# CREATING AUGMENTED REALITY AUTHORING TOOLS INFORMED BY DESIGNER WORKFLOW AND GOALS

A Thesis
Presented to
The Academic Faculty

by

Maribeth Gandy Coleman

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in Computer Science in the
School of Interactive Computing

Georgia Institute of Technology
December 2012

**CREATING AUGMENTED REALITY AUTHORING TOOLS INFORMED BY DESIGNER WORKFLOW AND GOALS**

Approved by:


Dr. Blair MacIntyre, Advisor
School of Interactive Computing
*Georgia Institute of Technology*

Dr. Keith Edwards
School of Interactive Computing
*Georgia Institute of Technology*

Dr. Elizabeth Mynatt
School of Interactive Computing
*Georgia Institute of Technology*

Dr. Jay David Bolter
School of Literature, Communication, and Culture
*Georgia Institute of Technology*

Dr. Beki Grinter
School of Interactive Computing
*Georgia Institute of Technology*



Date Approved: September 21, 2012

*To my husband*
*For inspiring me to be a better version of myself for 20 years and counting*

## ACKNOWLEDGEMENTS

First, I must express my utmost appreciation to Blair MacIntyre. He is not only my advisor, but also my mentor and friend. He introduced me to AR and taught me how to be a researcher. He has always treated me like a peer, when he did not have to. He encouraged me to pursue the PhD. in the first place, and then put up with me as a non-traditional student for almost a decade. If I had anyone else as an advisor I probably would not be completing this degree.

I also wish to thank my committee for their time and support. I realize I have gone about this journey in a strange and circuitous way. I greatly appreciate their patience and encouragement.

Dr. Anne McLaughlin is my oldest and dearest friend. I am lucky that she is also a wonderful researcher and educator. When this degree seemed its most unlikely, Anne calmly but resolutely provided every bit of help she could. I aspire to be as amazing as she is one day.

I also want to thank my dear friends Ashley Sasnett, Susie Tamasi, and Fran Walsh. The support these great women provided in the final weeks was invaluable.

I learned to value education and scholarship from my lovely family, Maurice, Janice, and Megan Gandy. My parents made incredible sacrifices for my education. No words can fully express how much that sacrifice means to me.

Lastly, my husband, Irwin Coleman, and I have been on this journey together since AP computer science class in 10$^{th}$ grade. All that I am as an adult is due to his encouragement and love.

**TABLE OF CONTENTS**

## LIST OF FIGURES

## LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| AR | Augmented Reality |
| MR | Mixed Reality |
| VR | Virtual Reality |
| HCI | Human Computer Interaction |
| VRML | Virtual Reality Markup Language |
| HMD | Head-mounted Display |
| IDE | Integrated Development Environment |
| PIC | Peripheral Interface Controller |
| POV | Point of View |
| DART | Designer's Augmented Reality Toolkit |
| TAM | Three Angry Men |
| FAM | Four Angry Men |
| VOO | Voices of Oakland |
| TUI | Tangible User Interface |
| IDE | Integrated Development Environment |
| VRPN | Virtual Reality Peripheral Network |
| AEL | Augmented Environments Lab |
| SL | Second Life |
| ARSL | Augmented Reality Second Life |
| MMO | Massively Multiplayer Online |
| WOz | Wizard-of-Oz |

| | |
|---|---|
| HTML | HyperText Markup Language |
| GUI | Graphical User Interface |

## SUMMARY

In a 20-year period, AR has gone from being viewed as a heavyweight technology, only appropriate for industrial and military applications, to a new medium for a variety of commercial and artistic applications. As a result there has been an increasing need for tools to support AR experience design and development that fully address the needs of non-technologists. From my AR research, I learned that three critical components for these authoring tools are support for an established content pipeline, rapid prototyping, and user experience testing. The history of media teaches us that AR also shares underlying technologies with a variety of more mature media such as games, VR, film, and the web with existing workflows and tools, which are used by a variety of creators. Therefore, we created an AR authoring tool that supported these three critical components, and whose design was informed by established approaches in these related domains, allowing developers with a range of technical expertise to explore the AR medium.

In this dissertation I present four main contributions. The first was an exploration of the AR design space to investigate "big ideas" in the field, focused on close collaboration with designers. This work resulted in guidelines for AR authoring tools, and informed the development of the Designer's Augmented Reality Toolkit (DART). The design of DART emphasized the needs of developers outside the AR research. These guidelines were validated via internal and external projects. A qualitative study of long term DART use that provided insight into the successes and failures of DART as well as additional understanding of AR authoring needs. Lastly, I trace two main threads to highlight the impact of this work, the development of the AR Second Life system and the creation of the Argon AR web browser.

# CHAPTER 1

# INTRODUCTION

Augmented Reality overlays virtual content, such as computer generated graphics, on the physical world (Azuma, 1997). The augmented view of the world can be presented to the user via a head mounted display, a tablet/mobile device, or projection on the physical space around the user. While Ivan Sutherland first presented the concept of the "Ultimate Display" in 1965 (Sutherland, 1965), it was not possible to truly implement augmented reality applications until almost 25 years later (Caudell & Mizell, 1992). Therefore, the field of AR research is usually considered to have begun in the early 90's. In this 20-year period, AR has gone from being viewed as a heavyweight technology, only appropriate for industrial and military applications, to a new medium for art, games and entertainment applications. Evolution of the field is due in part to the extensive research that has gone into exploring the AR application space, but also the recent rise of powerful mobile devices that make it easy to deploy a wide-variety of AR applications to consumers.

This is a critical moment for the field of AR. Over the past three years, AR technology has become accessible outside of computer science research labs. At first this mainly HCI researchers, but now we see participation from a variety of groups including visual and performance artists, user experience experts, game developers, marketing professionals, toy designers, web developers, entrepreneurs, etc. As a result, there is an increased demand for tools to support AR experience design and development that fully address the needs of these non-technologists.

My early work highlighted the importance of creating tools informed by the actual workflow and goals of non-technologists and designers and have spent the last decade of AR research developing and studying such tools. Low-level AR research in computer

vision, graphics, sensors, and optics is, of course, critical to the success and growth of AR. However, AR is becoming a mature research field made up of all types of researchers and practitioners, and the work presented in this dissertation has focused on making these technologies accessible to a diverse audience, with the goal of accelerating their adoption and evolution.

This work has relied heavily on collaboration with researchers outside of AR, computing, and the academy. From the beginning, I have explored the AR application space, informed by these collaborators, leveraging their domain expertise, and designing to meet their needs. Based on early experiences creating AR applications, developing authoring tools and evaluating these applications and tools, I learned that critical components for the advancement of AR were accessible authoring tools that support an established content pipeline, low-level hardware access, rapid prototyping, debugging, and the types of applications non-technologists want to create. In this dissertation I will demonstrate how, through this focus on designer workflow and goals, I have developed authoring techniques and created tools that have been used for a wide-array of applications and have contributed to the advancement of AR from a niche technology to a field of research and commercialization.

The theme of this dissertation is creator and process-centric design with the ultimate goal of stimulating media developers to engage with the AR medium. While many of my AR colleagues have built tools for programmers, this research thread is in the tradition of work by researchers such as Landay, Klemmer, and Li, who examine designers and their process to inform an ecosystem of tools that support activities such as the creation of web sites (Klemmer, Newman, Farrell, Bilezikjian, & Landay, 2001), location-based applications (Li, Hong, & Landay, 2007), tangible user interfaces (Klemmer, Li, Lin, & Landay, 2004), and online advertisements (Dow et al., 2011). My research, like theirs, is at the intersection of the design and development process.

## Thesis Contributions

The contributions of this dissertation are:

1. **Analysis of creator workflows and authoring needs via collaborative projects that explored the AR design space while resulting in deployable systems.** I participated in the design and development of four early applications to explore "big ideas" in the field of AR: continuous location-aware audio content delivery, gaming & entertainment, enhancement of live performances, and interactive dramatic narratives. These applications were bounded by the limited technology of the time but were designed to be deployable within these constraints. This approach allowed us to study the application space with real users and experiment with what the field of AR could be when technology and authoring capabilities matured. The applications presented here are particularly interesting in that they included collaborators from outside of AR research including designers, performers, media theorists, and writers. The applications were informed by the needs of these collaborators and guided by their artistic or research goals. Three impactful themes emerged regarding AR authoring: the significance of *approachable access to low level hardware and sensors*, the importance of *a mature content pipeline*, and the need to support *participation by a variety of stakeholders* in the process. These projects also revealed the effect a lack of support for iteration and prototyping has on the quality of AR applications and the power of leveraging existing media development tools in AR authoring. This work informed subsequent tool building as AR technology matured, guided not by AR research goals but the needs of non-technologists.

2. **Development of an authoring tool that allowed creators to engage directly with the AR medium, leveraging familiar metaphors and tools, while providing features that addressed the authoring challenges unique to AR.** I was the lead developer of an authoring tool for AR, called the *Designer's Augmented Reality Toolkit* (DART) that was focused on the needs of developers outside of the AR research community, informed by their workflows and goals. This tool leveraged an existing commercial tool already used by our target demographic to develop interactive media applications. Previously, we had implemented AR applications for our non-technologist collaborators; the goal of DART was to allow them to work on their own.
    a. **Development of guidelines for AR authoring tools.** The goal of DART was to not merely create a software artifact, but to use the research and development process to generate guidelines that would inform future AR authoring tools. We identified a variety of developer requirements unique to AR authoring including the need to: *design and test in situ*, allow for *seamless switching of sensor/tracking solutions*, support the *capture and communication of early design ideas between stakeholders*, and *prototype the entire user experience* (virtual and physical).
    b. **Validation of the guidelines.** A collection of internal and external projects developed with DART provided information regarding the utility and shortcomings of the tool. An examination of the design process for these applications helped to validate our AR authoring guidelines and the value of the DART environment.
3. **A study of long term DART use.** I performed a qualitative study with a population of eight external DART developers to capture their experiences using the tool in large-scale projects. The developers represented a spectrum of technical expertise from a computer science researcher (computational thinker) to

a theater director (non-computational thinker) and project goals (HCI experiments to art installations). This study yielded reflections on the utility of DART and additional findings related to the challenges and needs of AR authoring in general. Key insights include the challenge of *debugging* AR systems, the demand for *access to hardware systems*, the need to convey the *affordances and constraints* of AR to developers, the impact of *rapid prototyping support*, the importance of *developer communities*, the requirements for *effective collaboration among diverse teams*, and the value of a *layered authoring environment*.

4. **Influence on subsequent authoring tools.** The previous contributions led to a new era of projects in our research group that are based upon a tradition of collaboration between diverse teams, including artists, media theorists, game designers, performers, architects, historians, web developers, marketing professionals, and entrepreneurs. The theme of these projects is to design and evaluate applications addressing the "big ideas" in AR, which the research community began investigating 20 years prior, guided by modern software and hardware technology, web, and social network architecture and the artistic, commercial, and research interests of our collaborators. Two main threads illustrate the impact of my previous contributions. DART revealed the value of an AR authoring tool, emphasizing the performative aspects of AR applications over programming power, and inspired the development of the *AR Second Life* system. ARSL then led to a series of explorations into "mirror worlds", inspired by our early work in application building and tool development, culminating in an initiative to create an AR web browser called *Argon*. Argon brings concepts from DART into the modern era by utilizing web technologies to support approachable authoring for non-technologists while making it possible to deploy applications on a large scale. Due to its web underpinnings, this project provides additional capabilities for massively multi-user systems and extensive data connectivity.

Argon is currently being used in a variety of AR development projects ranging from the artistic (e.g., tours of historic sites) to the commercial (e.g., delivery of AR technical instructions).

## Dissertation Outline

The remainder of this dissertation begins with a summary of the background and related work in three domains of AR research that mirror the progression of work over the past ten years: application building, authoring tools, and deployment. The subsequent five chapters present the previously listed contributions in detail. Chapter 3 describes the four projects of the early application-building era, highlighting aspects that comprise Contribution 1. Chapter 4 contains a detailed discussion of the DART architecture and the authoring tool guidelines that resulted from its design and development and then, in Chapter 5, nine diverse projects developed with DART are presented that provide validation of the tool and guidelines (i.e., Contribution 2). Chapter 6 presents Contribution 3, a qualitative reflection upon DART and AR authoring needs gathered via interviews with a community of external DART developers and AR experts. Lastly, Contribution 4 is discussed in Chapter 7, which presents our current AR research that is the culmination of our decade long inquiry into the AR design space, authoring tool requirements, and the needs of non-technologists. I conclude with Chapter 8, which summarizes the contributions.

# CHAPTER 2

# RELATED WORK

Over the past 20 years Augmented Reality research has gone through several distinct eras. Early application building explored the design space and initially demonstrated the types of systems that we are still striving to make a reality today. This work then informed the creation of authoring tools to support rapid prototyping and development by people outside of AR research. Presently, we are in the era of deployment, where the research community is working to create deployable platforms, scalable architectures, and applications that run reliably on consumer hardware. We are drawing on the application design, tool building, system engineering, and HCI research of the past to create real systems connected to live data.

This dissertation work, identifying and designing tools for the AR designer's needs, must be done in the context of related research in the AR and HCI communities. Therefore, in this chapter, I discuss the significant work in each of these AR research eras: application building, authoring tools, and deployment.

## Application Building

While the AR community continues to engage in application building, the work that went on in the early years of AR research was particularly influential. Early researchers explored AR's big ideas via prototypes, while still bounded by the reality of technology at the time. At this point, AR researchers were struggling to build prototype systems exploring the application space, exposing AR to "the wild." In their (and our) applications the goal was to explore the "low hanging fruit" concepts for AR, even though "the branches were high."

Throughout the 1990's and early 2000's, AR development was onerous and only limited deployment was possible. The systems often relied on custom and specialized

hardware and software. Application development used low-level tools such as C++, OpenGL, and VRPN. Later, some were developed with higher-level tools including 3D game engines. Often we created demonstration systems, physical installations, prototypes, and systems that needed careful oversight while in use, but they made it possible for researchers and a limited number of users to experience live interactive AR systems. The resulting systems demonstrated the "big ideas," but they could not be deployed into the world yet and they usually did not have connections to live data sources.

Azuma's "Survey of Augmented Reality" (Azuma, 1997) provides a snapshot of the state of AR in the early to mid 1990's. At this time, most AR research fell into one of six categories: medical, manufacturing and repair, annotation and visualization, robot path planning, entertainment, and military aircraft. One of the first AR systems was designed to guide a technicians in the building of wiring harnesses that form part of an airplane's electrical system (Caudell & Mizell, 1992). Feiner et al.'s KARMA system provided AR instructions overlaid on a printer (Feiner, MacIntyre, & Seligmann, 1993), while Rose et al.'s system supported a user's understanding of physical automotive parts by providing AR annotations (Rose et al., 1995). These were the first explorations of what would become a very active area of AR research. Early projects revealed the promise of AR support in an industrial setting, motivating subsequent work that has since proven that industrial AR can improve the speed and accuracy of workers (Henderson & Feiner, 2011; Tang, Owen, Biocca, & Mou, 2003).

The use of AR to annotate the real world and to visualize data in situ was also of great interest to researchers. As with industrial systems, medical applications were a domain where the application of AR was particularly appropriate due to the need to consult visual data, such as ultrasound, pertaining to a patient during procedures. A virtual fetus was visualized within a pregnant patient (State et al., 1994) and, later, AR was applied to a needle biopsy procedure (State et al., 1996). This continues to be an

active area of AR research. For example, Okur et al. are currently evaluating an AR system deployed in operating rooms (Okur, Ahmadi, Bigdelou, Wendler, & Navab, 2011).

The use of AR annotations was also applied on the macro scale. Feiner et al developed the "Touring Machine" system to provide outdoor information about buildings and locations (Feiner, MacIntyre, Hollerer, & Webster, 1997). This work revealed many of the challenges of information organization in AR and led to research in the UI management of large numbers of AR annotations (Bell & Feiner, 2000); a problem of increasing significance in the modern era of AR where commercial applications are attempting to present large amounts of live data to users. An alternative approach to an entirely visual presentation of augmentations is the use of the audio channel. Bederson created an early audio AR tour guide (Bederson, 1995), which informed my own exploration of audio AR a few years later (see Chapter 3).

During the initial application building era the entertainment domain was a less active area of research. However, Maes developed the ALIVE system in 1995, which used a projection-based "magic mirror" style presentation to provide an edutainment experience for children involving intelligent virtual characters (Maes, 1995). More recently, this concept has emerged in commercial products, including the Sony EyePet ("Sony EyePet," 2011). Later, Cheok et al. created an outdoor version of the game Pac Man. One player takes on the role of Pac Man, collecting virtual power pellets around the environment while pursued by other players in the role of ghosts (Cheok et al., 2004). Thomas et al. modified the popular first-person shooter computer game, "Quake," to create an outdoor AR version of the game (Thomas et al., 2000). AR Quake is of particular relevance to this dissertation as it was an early example of researchers leveraging an existing professional platform and content pipeline for AR authoring.

## Authoring

The exploration of application domains prompted AR researchers (and researchers in other emerging domains such as ubiquitous computing) to develop authoring tools. Many AR researchers created tools to support their AR research goals related to tracking (Billinghurst, Bowskill, Jessop, & Morphett, 1998), distributed systems (MacIntyre & Feiner, 1998; MacWilliams et al., 2003), outdoor applications (Piekarski, 2006; Piekarski & Thomas, 2003), user interfaces (Dunser, Grasset, & Billinghurst, 2008), CSCW (Billinghurst, Weghorst, & Furness, 1998), and mobile AR (Schmalstieg & Wagner, 2007). The field was at a point that more sophisticated applications were possible, but this level of development required the ability to rapidly prototype, iterate, and deploy. The component that was severely lacking in the previous application building era was this ability to evaluate ideas early and often via rapid prototyping and to do iterative design. In our own work it could take six months to a year to develop an application and it was often difficult to evaluate the application at all before then.

Technological advancements in the late 1990's and early 2000's also played a role. Early on, researchers had envisioned higher-level tools, but most software development in general was low level and difficult until the late 1990's. The rise of the World Wide Web produced a class of high-level authoring tools like Macromedia Director, Adobe Dreamweaver and scripting languages such as Perl. People other than computer science professionals were developing applications. Graphical IDEs, such as Microsoft Visual Studio, and GUI toolkits, such as Microsoft Foundation Classes, and Java Swing that addressed the need for tools and APIs that catered to GUI-centric application development began to appear. At the same time, 3D graphics capabilities were becoming mainstream and mature. For example, the VRML standard was a file format for 3D graphics designed with the World Wide Web in mind ("VRML Virtual Reality Modeling Language," 2011). Non-technologists were modifying 3D games, such

as Quake, via tools exposed by the developers. As a result, the AR researchers now had tools to leverage for AR development.

Much of the work in the AR community has focused on improving tracking and display technology. There have been a number of AR authoring tools developed over the years with these goals in mind, such as Studierstube (Schmalstieg et al., 2002) the ARToolkit (Billinghurst, Bowskill, et al., 1998), Tinmith (Piekarski & Thomas, 2003), Coterie (MacIntyre & Feiner, 1996), and DWARF (MacWilliams, Reicher, Klinker, & Bruegge, 2004). All of these tools require the application developer to work at a fairly low level, with languages like C or C++. Studierstube did leverage OpenTracker, which eased requirements of hardware access by providing a method of defining device configurations via XML (Reitmayr & Schmalstieg, 2001). Although the Tinmith system required low level programming, Piekarski did use the platform to create a variety of in situ modeling interfaces that allowed a user to quickly create 3D models outdoors through pointing and hand gestures (Piekarski, 2006). However, none of these systems attempted to provide a designer-focused prototyping environment, but instead were targeted at AR technologists. For example, Coterie, a prototyping environment that provided language-level support for distributed virtual environments, was used to create some of the early AR applications such as the Touring Machine (MacIntyre & Feiner, 1996, 1998).

Of these, the most widely used is the ARToolkit, both because the technology requirements for using it are modest (i.e., a webcam and printed markers) and because it is relatively easy to create content using VRML. ARToolkit was combined with a commercial real-time 3D environment called Touch Designer, which artists and designers can use to create interactive augmented-reality environments (Berry et al., 2008). ARToolkit was also ported to the Adobe Flash environment, allowing developers to easily deploy marker tracked AR applications via a web browser. This FLARToolkit ("FLARToolkit," 2011) attracted a community of interactive media designers and was

used for several marketing campaign applications including GE's SmartGrid (lester, 2011). COMPOSAR was a rapid prototyping built using the ARToolkit that supported visual programming as well as scripting and an immediate mode for run-time testing (Seichter, Looser, & Billinghurst, 2008).

Subsequent projects confronted authoring from the other end of the design spectrum, creating environments to simplify the creation of AR systems with minimal programming. APRIL (a scripting environment built on Studierstube) was an XML-based language for authoring AR (Ledermann, 2011). AMIRE, completed and evaluated in 2004, was focused on creating graphical authoring tools for specific AR domains, rather than a general-purpose environment. While certain applications could be more quickly developed in this tool, the designers were constrained to problems that the tool was designed to solve (Abawi, Dorner, Haller, & Zauner, 2004; Grimm et al., 2002). CATOMIR, built on top of AMIRE, was a graphical authoring environment that used a dataflow approach, where their system components were hooked together with "wires" (Zauner & Haller, 2004). ImageTclAR was a Tcl/Tk-based development environment designed to provide basic AR system functionality for application developers with different backgrounds and programming abilities from relative novices to experienced systems developers (Owen, Tang, & Xiao, 2003). MARS (Mobile Augmented Reality System) was designed to support mobile user interface studies in AR and provided a comprehensive set of reusable user interface components
(Hollerer, Feiner, Terauchi, Rashid, & Hallaway, 1999). The MARS 3D graphical user interface was used by non-programmers to create and edit situated documentaries (Guven & Feiner, 2003).

In the past few years, researchers have begun leveraging games and game engines for their AR authoring. Oda et al. developed Goblin XNA, a set of AR specific components added to the Microsoft XNA engine (Oda, Lister, & Feiner, 2008). In the commercial realm, Qualcomm has released their AR library, Vuforia, as a plugin to

Unity, providing video access and natural feature tracking to game developers ("Qualcomm Augmented Reality SDK," 2011).

Other relevant authoring work can be found outside the AR community. The Context Toolkit was an infrastructure to support context aware applications that provided developers with easy access to contextual information and operations to manage it (Dey, Salber, & Abowd, 2001). The Phidget toolkit (Greenberg & Fitchett, 2001) was aimed at making tangible devices available to designers, and solved some fundamental problems of prototyping. This work was a key inspiration for our development of DART and support for Phidget prototyping was eventually integrated into our DART authoring environment (MacIntyre, Gandy, Dow, & Bolter, 2004). There have been numerous research systems created over the years to support the exploration of new electronic media by novice programmers with an eye toward developing better interface and programming technologies. One that is similar to our work is Alice (Conway et al., 2000), although it was focused on lowering the threshold of entry into the world of 3D graphics and VR programming, rather than explicitly supporting early design activities.

Our work has been heavily influenced by the work of Landay, Klemmer, and Li in supporting a variety of design and prototyping activities. Their work, like ours, often focuses on enhancing existing work practices. For example, in the Designer's Outpost project they found that web designers often used pens, paper, walls, and tables during the early phases of design (Klemmer, et al., 2001). This inspired them to create a tangible user interface that combines the affordances of paper and large physical workspaces with the advantages of electronic media to support information design. The motivation for our use of sketch-based content for early user experience testing also comes from the work of Landay and his collaborators at CMU and Berkeley (Landay & Myers, 2001). The Wizard of Oz components in DART were inspired by "Suede," which supported informal prototyping of speech interface (Klemmer et al., 2000) and by Li et al.'s WOz prototyping of location-based applications (Li, et al., 2007). There are many similarities

in approach between DART and Papier-Mâché, which was an open-source toolkit for building tangible interfaces using computer vision, electronic tags, and barcodes. Informed by structured interviews with TUI experts, Klemmer found that building these UIs required "getting down and dirty" with input technologies such as computer vision and, as a result, few people could develop them. Therefore, the goal of Papier-Mâché was to make TUI development accessible to a greater number of developers. It also provided a high-level event model for working with these technologies that facilitated technology portability (Klemmer, et al., 2004). More recently, Dow et al. have been studying designer workflow with the goal of improving the prototyping process. For example, they have found that sharing multiple designs during the prototyping process improved exploration, group rapport, and results (Dow, et al., 2011).

## Deployment

In the past several years there has been a growing interest in AR research, which is due to certain advancements that have made wide-scale deployment of AR applications possible. With the rise of mobile devices and the increasing availability of geo-located data, AR has recently become a viable platform for commercial interests. A variety of commercial games and marketing applications have been released including an application from GE promoting their SmartGrid technology (lester, 2011), AR baseball cards from Topps ("TOPPS 3D LIVE Online Augmented Reality Trading Card App," 2011), Nintendo's 3DS AR games (McFerran, 2011), and Sony's AR enhanced collectible card game "Eye of Judgment" (Wikipedia, 2011). This is the first opportunity AR researchers have had to create systems that are deployed to thousands of users that utilize vast amounts of data available via the cloud. There are numerous projects and products focused on creating live versions of the applications the pioneers started researching 20 years ago. The increasing sophistication of the World Wide Web, and Web 2.0, in particular, have resulted in powerful ubiquitous cloud services exposed via

easy-to-use APIs and scripting languages. Also, millions of people are already using these services and they are seeded with rich information. The ubiquity of mobile devices, with their built-in location services, has resulted in vast geo-located data sets as well. Many promising AR concepts from the past relied on hypothetical cloud services that would have had to be written from the ground up or on knowledge of the environment (e.g. street data, topographical data, 3D models of buildings, etc.) that was not widely available. Now they are often easily implemented using sophisticated authoring tools and the World Wide Web.

Many people are now developing AR applications. Some are computer scientists, but many are graphics designers, game developers, marketing professionals, toy designers, web developers, entrepreneurs etc. The academic community is still contributing, but breakthroughs are coming from other groups now as well. This has implications for our current research work with non-technologists and professional tools. We must engage with and support these new practitioners and seek to solve the problems of scale and distribution associated with live AR systems. Argon, the AR web browser, which is discussed in Chapter 7, exists in an ecosystem of current initiatives, both academic and commercial, that are currently working to bring the web into the physical world via AR.

The concept of bringing the information from the digital world and the Internet into the physical world has long been explored. "Windows on the World" incorporated an existing 2D window system within a 3D virtual world (Feiner, MacIntyre, Haupt, & Solomon, 1993). This system took XWindows windows from the desktop and placed them into the physical world. This 2D information could be linked to the HMD, to a surrounding information sphere, and to locations and objects in the world. It was a precursor to the modern consumer mobile AR applications that have recently become available (e.g., Layar, Junaio, and Wikitude). WorldBoard proposed a planetary augmented reality system that would provide innovative ways of associating information

with places (Spohrer, 1999). Spohrer envisioned a system that would allow users to post content (from pictures to text) on any of the six faces of every cubic meter of space on the globe. The goals of the Real World Wide Web (RWWW) project were very similar to those of Argon and the issues they foresaw related to presentation and user interface are relevant to our current work (Kooper & MacIntyre, 2003). The vision of RWWW of an outdoor, GPS tracked, mobile applications superimposing data from the World Wide Web on the user's surroundings. Kooper et al. developed a prototype "browser" that allowed them to experiment with interfaces to this 3D spatialized information space. They were interested in exploring the implications of adding context information to documents on the World Wide Web. They noted that there was a wide range of research to be done, as the interface design must balance the conflicting requirements of minimizing the volume of information displayed (to avoid distracting the user and cluttering their visual field) with the need to provide rich context (to capitalize on the users ability to rapidly scan and synthesize data).

In the last three years, with the advent of mobile phones with GPS, 3D graphics capabilities, data connections, and application distribution channels, there has arisen a crop of commercially available AR platforms most of which are designed for outdoor information browsing and retrieval. Wikitude released their Wikitude World Browser ("Wikitude,"), which presents location-based Wikipedia and Qype content. Layar distributes the Layar Reality Browser ("Layar,"), which allows developers to create custom "layers" of information that can be served up to users via their custom publishing platform. Metaio has a markerless tracking solution as well as authoring tools ("Metaio | Augmented Reality 3D," 2011). Their Unifeye Design 2.0 supports the creation of presentations and live-marketing via a GUI interface. They have also created the Junaio mobile AR browser, which has an open API and allows the developer to define applications via XML. Their Creator system provides a drag and drop interface for authoring. Total Immersion’s D’Fusion Pro lets developers access high definition video,

multi-camera systems, face recognition/tracking services, markerless tracking, and the Microsoft Kinect to author performance, industrial, and presentation applications ("Augmented Reality Software and Solutions by Total Immersion | Augmenting Your Reality," 2012). Their Studio software provides a GUI interface for importing 3D content, creating a tracking scenario, and adjusting the content relative to the marker.

# CHAPTER 3

## APPLICATION BUILDING AND EXPLORING BIG IDEAS

Prior to my work on DART, my AR research was focused on demonstrating the range of possibilities for augmented reality. The actual implementations were bounded by the state of technology and tools, but the motivation was to explore the future of the AR medium and was informed by collaborations with non-technologists. This chapter will discuss four projects that illustrate the trajectory of my work in this phase, and how they influenced this dissertation work. These four projects have the common theme that they were collaborations with non-technologists and/or people from outside the AR research domain and, while these were research projects, they were designed for deployment. As a result, the design decisions were informed by practicalities of content pipelines (primitive as they were), the aesthetic choices of our collaborators, and the realities of technology at they time. There was limited support for authoring; the development was time intensive and at a very low level. There was a gulf between the tools, the technological realities, and our collaborators. It was difficult for the non-technologists to grasp the capabilities of the primitive technology and the lack of any tools that would allow them to prototype or engage with the medium directly made it challenging for them to develop creative strategies for designing within the constraints. However, these projects allowed me to investigate existing workflows and tools used by creators in the media of live music performance, theater, film, and sound design.

I believe it is useful to reflect upon these projects because with our own work, unlike other "first era" AR, we have full knowledge of our motivations and ideas. I can provide a unique perspective on the entire design process that was involved, even the less successful approaches. When reviewing this early AR work it must be noted that, while the ideas were sophisticated, the actual artifacts that resulted were constrained by the

primitive technology and authoring tools of the time. We were attempting to identify the nexus where technology could meet concept and research goals. The important contribution is often the "big idea" that was at the core of the work. People are still exploring these ideas today, and many are still a ways away from appearing in the commercial domain.

Three common themes from all four projects emerged that became key influences upon our subsequent authoring tool design: the significance of *approachable access to low level hardware and sensors*, the importance of *a mature content pipeline*, and the need to support *participation by a variety of stakeholders* in the process. While a variety of findings resulted from each project, this chapter focuses on highlighting those three core concepts.

### Audio Augmented Reality: Guided by Voices

This first project was a simple audio-only AR platform. The Guided by Voices system was an example of designing a deployable user experience within the very limited bounds of mobile technology at the time. One goal of the project was to create a platform that could be deployed to large numbers of users at events. There were no mobile systems at the time that were low cost, low power, and small enough to enough to meet this goal. Therefore, we decided to pursue a lightweight system that could present location-based auditory augmentations. Informed by the Audio Aura (Mynatt, Back, Want, & Frederick, 1997) and Bederson's audio AR tour guide (Bederson, 1995) systems, Guided by Voices also delivered location-based non-headtracked/non-spatialized audio clips, but our system utilized an even lower capability mobile platform, a Rio MP3 player. For position tracking Dr. Thad Starner and his students developed a low-resolution indoor tracking system using RF transmitters and receivers. This system supported room level accuracy, but performed more robustly in larger spaces. The MP3 device could be controlled via the headphone port. Therefore we developed custom circuitry controlled by a PIC

microcontroller that allowed us to programmatically choose tracks to play (see Figure 1. ). The game logic consisted of a state machine programmed on the PIC that took as input the current location identifier and output the desired track. This project was successful due to our creative and novel use of hardware. However, the slow and tedious development process affected the sophistication of the resulting applications. Therefore, this project highlighted the need for high level programming tools that would allow *access to arbitrary hardware*.

**Figure 1.** The Guided By Voices hardware and packaging: Rio MP3 player, receiver, and PIC

We used this platform to create two prototype games and a tour guide. The first, Assassin, was created with input from students in the Mobile and Ubiquitous Computing course in the College of Computing. In this game participants would not only carry the Rio device, but they also wore the RF transmitters. In this way the system could detect which "assassins" were in the surrounding area and audio cues could warn the wearer of the system. If an assassin stayed within range for a set length of time he/she would

receive a “kill.” The second game, a fantasy role-playing game, tasked the player with saving his elven friend by physically visiting locations to collect weapons and battle foes (Lyons, Gandy, & Starner, 2000). For this project we collaborated with some members of the GVU AudioLab to create the soundscape. The GBV platform was also used to create a tour guide for the Beware Home (which is described fully in the next section) (Kidd, Starner, Gandy, & Quay, 2000). Unlike the fantasy RPG, this experience was meant to entertain, but also to provide information on a variety of research demonstrations and to guide visitors through a space. The team for the BewareHome project included a variety of people including computer scientists, graphic designers, sound designers, scriptwriters, and electrical engineers. The collaboration that created the audio tour guide included LCC students who wrote scripts, sound designers that crafted the soundscapes, and technologists that instrumented the house based on the tracking requirements defined by the experience design.

The development tools were extremely low-level, but the overall concept was easy to communicate to all the stakeholders of the three projects and therefore, their disconnection from the tools and their lack of opportunity to work directly with the technology was not a major impediment. Although simple, an *effective content pipeline* was devised using a sound-editing program to create sound elements and to preview the user experience in a rough way.

The main challenge in collaboration was explaining that different location identifiers could trigger different sounds based on what other areas had already been visited. The applications were based on the players moving through a space, visiting various hotspots. At a hotspot they would hear a vignette telling them what transpired there. A player might locate an item or meet a character. The state machine allowed us to keep track of which locations had been visited and, thus, have game logic that responded accordingly (see Appendix A). For example, in one location in the RPG the player would encounter a goat which would follow him through the rest of the game, if the player went

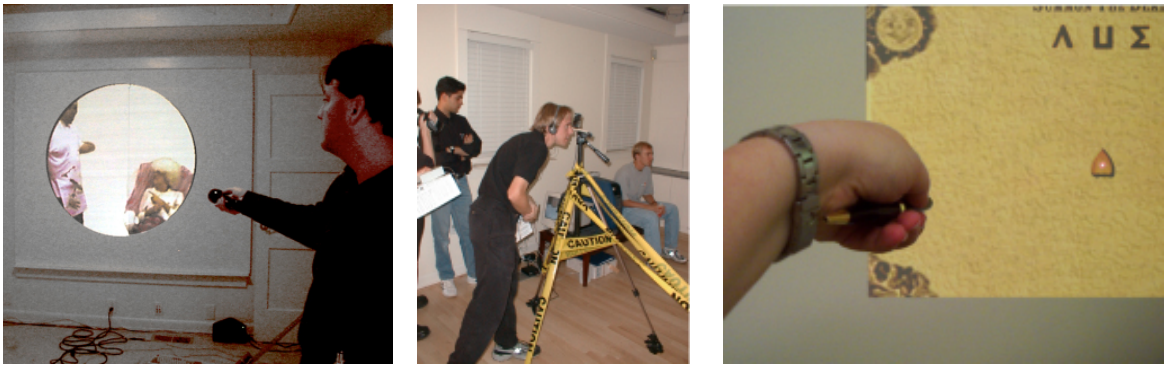
to the location of the troll, the goat would make noise, waking the troll, and causing the player's death. This action was portrayed entirely through sound. We adopted a philosophy of sound design that was based on every vignette having three layers: an ambient sound layer that described the environment to the player (e.g. the wet dank echoing sound of a cave), action sounds that conveyed what happened at the location (e.g. a dragon roaring, belching out a fire ball, and a human scream as his target turned to ash), and a narrator who would explain clearly what had just happened (e.g. "I'm sorry. You have died. You got burned to a crisp by the dragon. Too bad you didn't have a shield to protect yourself. Please begin your game again"). The stakeholders, no matter their technical expertise, easily understood this application logic and, thus, *the diverse team was able to collaborate effectively* to craft creative ways to use this simple input/output mechanism to make the game world dynamic.

This project was representative of AR projects of the era, with its use of custom hardware and very low-level development. It was made possible by hardware innovations including the release of the first mp3 player and the availability of RF transmitters and receivers. The applications had to be designed around a host of constraints, but we were able to create compelling applications mainly due to our ability to iteratively develop content with little difficulty and due to the fact that our diverse collaborators were able to easily grasp the potential, and the constraints, of GBV applications. We were also able to have real users interact it without, literally, hovering over them or controlling it for them. The system was deployable enough that we were able to get feedback "in the wild."

### Creating AR Installations: The BewareHome

The goals of the Beware Home were similar to GBV. We designed a suite of AR demonstrations that were robust enough to be placed in a live exhibit (Kidd, et al., 2000). The BewareHome was a haunted house treatment of AwareHome research. The goal was to highlight computing research while providing an entertaining experience for visitors. It

was opened to hundreds of visitors in October of 2000. The exhibit was the product of work from a large team of collaborators spanning multiple research labs and colleges on Georgia Tech's campus. The team included research scientists building hardware, designers developing the overall story that was told via decoration, sound effects, costumes, and technology, and graduate students attempting to rework their research applications to fit within the overarching narrative. The installations included five types of AR presentation: an HMD-based system in the kitchen, a projection-based AR installation in the bedroom with laser pointer input, a "magic mirror" in the bathroom, a projection-AR experience controlled by a flashlight interface in the master bedroom, and the previously discussed audio tour delivered via Guided by Voices. This was an opportunity for us take advantage of the eclectic team assembled for this endeavor to develop creative AR experiences utilizing a variety of input and output modalities and observe real users interacting with them.




(a) (b) (c)

**Figure 2.** Beware Home exhibits (a) the flashlight-controlled AR system (b) a visitor views the virtual ghost through an HMD (c) a visitor "writes" on the wall with a tracked laser pointer

The five AR exhibits were:

- *A Ghost in the Kitchen*. The HMD exhibit allowed visitors to see a virtual ghost in the kitchen that would walk around and tell them a story about a murder that had occurred there (see Figure 2. (b)). The HMD was mounted to tripod for ease of viewing. This also simplified the tracking requirements, as only orientation was needed. In another room, the augmentations were presented via a projector.
- *Writing with Fire*. In another room, the visitors were prompted to write with "fire" on the wall via a vision-tracked laser pointer (see Figure 2. (c)). The visitors had to inscribe the correct magic rune on the wall to escape the room.
- *Magic Flashlight*. Upon entering the master bedroom, the visitors were presented with a magic flashlight that revealed residents of the past, acting out a scene on the wall (see Figure 2. (a)). An IR filter on flashlights allowed them to be tracked by camera. An advantage of this system was that multiple users could interact with it simultaneously.
- *Scary Mirror*. When visitors in the bathroom looked into the mirror they would see themselves along with frightening augmentations. In this case no real tracking was required.
- *Audio Tour Guide*. As previously discussed, GBV was used to provide a location aware tour that guided visitors through the space. This was an exploration of what is now a common idea, smart audio tour guides for museums and historic sites.

As with GBV, this diverse collection of AR experiences highlighted the power provided by *access to arbitrary hardware/sensors* (e.g. orientation sensors, HMDs, projectors, vision-based tracking, and face detection) for creative exploration of the AR medium, but also illustrated the challenges of achieving this access via low-level development tools (e.g. C++ and OpenGL). However, we were able to use Macromedia

Director for the two projection AR installations (i.e., Writing with Fire and Magic Flashlight), supported by a custom blob-tracking plugin that I developed. It was unusual at this time for a tool intended for designers to be used in this manner, but it allowed us to work more effectively with the graphic designer who was creating the content. She was skilled with Director scripting and it was her tool for choice for creating motion graphics. This was my first experience leveraging professional-grade media development tools for AR authoring. It revealed how the use of such an environment could supply an *effective content pipeline* and how it supported *diverse collaborators* with no AR experience, including the graphic designer, script writers, and the filmmakers, to contribute substantially to the creative process.

### Augmentation of a Live Performance: Duran Duran

This was one of the first AR projects where I worked very closely with collaborators far outside of traditional computing. We created a variety of augmented reality pieces for Duran Duran's live performances in the fall of 2001 (Pair, Chastine, & Gandy, 2002). Our design choices were informed by the needs of the performers and their artistic vision.

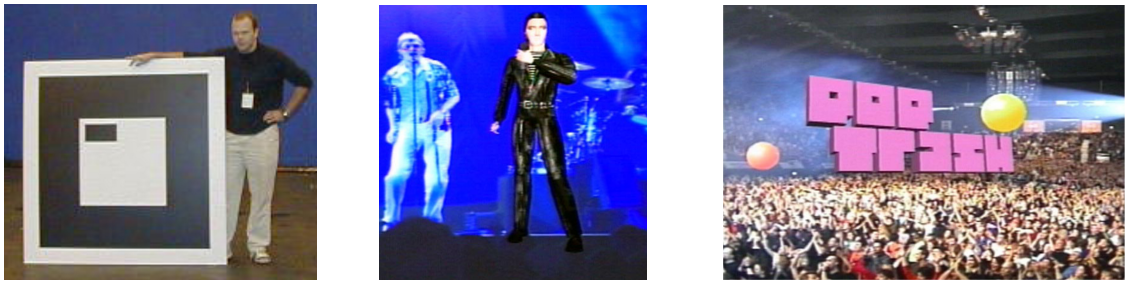
The process started with the band becoming excited by AR based on technology demonstrations the team showed them. The experiences were then created via a collaborative process with the band and crew. The design of the augmentations was informed by the themes and content of the songs they accompanied. Initially, the band provided a set list of songs to consider building effects around. The technical team then created early technology prototypes to start the brainstorming process and to explore technical solutions (e.g., blob tracking, image processing, fiducial tracking).

The demonstrations helped the band understand the technical constraints of the computer vision system, which was critical considering the harsh environment in which the applications would be used. Members of the band actually interacted with the demos

and saw first hand what environmental conditions could break the tracking. They developed a correct mental model of how the technology worked. Later, they would be frustrated by the limitations of the tracking, but they understood the source of those limitations.

To share ideas and progress updates, throughout the process we made videotapes of applications to show progress and produced a number of webpages with the band that showed screenshots, as content was developed iteratively. We were forced to develop a process that was *inclusive and approachable for the variety of stakeholders* including the musicians, managers, technical staff, 3D artists, and crew (to be discussed in Chapter 6).

The resulting experience was an unorthodox type of AR application, as the users (i.e. the band) could not see the augmented views and the audience did not have control over the viewpoint. Our team controlled the cameras; they were aimed at the stage and the augmented view was then projected on a large screen at the back of the stage. The AR vignettes relied on a variety of technologies. Some utilized registered AR (i.e., tracking was accomplished via large ARToolkit markers), two used blob tracking, and others were video effects. The vignettes included "predator vision" for the song "Hungry Like the Wolf", virtual cocaine rising up out of tracked hand, giant women dancing on stage for "Girls on Film", and projected AR on top of a tracked beach ball (with a retro-reflective cover) in the audience (see Figure 3. ). Not all of these were actually used in the live performances due to logistic and legal constraints.



(a) (b) (c)

**Figure 3.** Images of the Duran Duran Project (a) a large marker used for tracking on stage (b) a virtual Elvis performs with the band (c) augmentations on the crowd

Crafting promising concepts and working with a large and diverse team were not the only hurdles we faced in this project. We also had to address the challenge of *creating high quality content quickly*. The entire system had to be designed and deployed in a relatively quick time period for AR systems of the time (i.e., a few weeks). The development was tedious as it was based upon C++ and DirectX, but it was made possible by the recently released ARToolkit (Billinghurst, Bowskill, et al., 1998), which utilized the VRML format, which meant it was relatively easy to make small modifications to the models and the animations. However, it did not support sophisticated rendering. For this project, it was paramount that all the visuals be of a very high quality. The team's ideas had to be scaled down to those that could be built with VRML and guaranteed to work perfectly night after night on tour. We had to develop an art pipeline to create content for the tour, which was handled by a group of 3D artists in Slovakia. A significant obstacle was conveying to the artists the modeling requirements that were unique to this application (e.g., real-time rendering, models viewed from arbitrary POVs, design integration with physical stage elements) and achieving visuals of a level of style and fidelity that the band was comfortable with. Time was of the essence. The project leader commented, "They [the band] had a lot of great animation ideas. We just did not

have the time to do them. The 3d content creation process was something that they had to learn about."

This type of application and the development cycle might not have been possible a couple of years prior to the advent of the ARToolkit, which provided us with a portable tracking solution that could be easily set up at a new venue each night (see Figure 3 (a)). The system worked reliably and was used in the live performances throughout the band's tour. This project highlighted for us the difference between designing AR prototypes for the lab, versus creating robust systems that could be placed "in the wild", the importance of content and content development in such a project, and provided additional experience with tight collaborations with non-technologists.

## Dramatic AR Experiences: Three Angry Men

Three Angry Men (TAM) was another example of collaboration between AR researchers and non-technologists. It was an experiment in the use of AR to explore point-of-view drama (MacIntyre et al., 2003; MacIntyre et al., 2002). We were exploring how AR could be used to produce a different type of experience for the user. Our research questions included: Could AR be used successfully in an application that was more artistic than functional? What affordances of AR could enhance the dramatic experience? How would you design and create content for this type of presentation? The collaborators on this project included Dr. Jay Bolter, a professor in the School of Literature Communication and Culture, and two of his students. One of the students, Jeannie Vaughn, wrote her Master's thesis on the design of the dramatic AR experience. Emanuel Moreno was a design student, with Macromedia Director knowledge, who wrote the majority of the code for the system using that tool.

The experience was based on the classic film "Twelve Angry Men." In the story a majority of jurors are inclined to find the defendant guilty. One thoughtful juror changes

their minds regarding the defendant over the course of the experience by presenting well-formed and logical arguments supporting his innocence. The jurors are a diverse group and come to the deliberation with a variety of preconceptions and prejudices. Our AR application allows the user to experience the story from each juror's point of view.

The experience took place in a room with a table surrounded by three chairs. The user could sit wherever he wished. When his sat down he "inhabited" the character in that chair and would see the experience from their point of view and hear their inner thoughts. He was also free to get up and move to another seat at any time. The script and visual content was slightly different from each point of view, highlighting the differences in how individuals in a group of people perceive the same situation. The content consisted of 2D video rather than 3D models (see Figure 4. ). We believed that, although it was less realistic in some ways (i.e., the 2D video billboards would only look visually correct from one head position and orientation), the rich body language and facial expressions of the actors provided a more compelling experience. Due to various production and tracking limitations the actor billboards never lined up perfectly, but it did not seem to diminish the quality of the experience for the user.

**Figure 4.** A Three Angry Men user and the three virtual jurors

Initially, it was challenging to help the actors understand this new type of media. For various reasons, both creative and technical, we ended up shooting the video multiple times; as a result, one lesson we learned was how to explain the project to the actors. The actors were asked to act out each scene three different times, giving different performances in each. Doing three different versions of the same script was initially foreign to our actors. Although the words they spoke in each scene were the same, we asked them to change their tone, body language, and facial expressions to fit how they would have been perceived from the three different points of view, including their own.

The video shoot was a learning process for us as well as the actors. We realized that the camera (which would move depending on which point-of-view we were shooting) should be treated as the opposing jurors eyes. The actors needed to "break the 4th wall" and address the camera directly at times. They were encouraged to make eye contact with the viewer by looking directly at the camera. It was helpful for the actors to think of the physical presence of the camera as the head of the juror that would be sitting there and interact with it accordingly. Also, it was important to remember, for the script and for shooting, that you couldn't control where the user would be looking at any one time in the performance. Therefore, throughout the scene, all the actors needed to perform appropriate and meaningful body language, even if they were not speaking. It was these subtle cues that made the experience rich and effective at conveying the point-of-view differences (e.g., the African American juror gives the "bigoted" juror a threatening glance while the thoughtful juror is speaking). The actors were also encouraged to use their performance to help lead the viewer's focus to the pertinent juror through their body language and eye gaze. Also, similar to stage acting, we found that our actors needed to be quite broad in their portrayal; subtle movements and tonal changes were easily missed due to the mediating technology and the lack of a fixed viewpoint.

The user interface evolved throughout the project. There were three main UI decisions to be made: how the inner thoughts of the "inhabited" juror were conveyed,

how to allow the user to switch between points-of-view, and how, once the switch was made, to restart the experience from that point-of-view. To avoid the need for a separate input device and in an attempt to give the users a greater sense of presence through realistic physical interaction, we designed the POV switching to be based on which physical chair the user chose. If the user stood up and moved to another chair around the table the POV would switch to that of the juror that was assigned to that seat. We were concerned about users becoming confused if they chose to move around. We did not want them to miss part of the scene or become disoriented when they sat down to a slightly different version of what they had just been watching. Therefore, the act of standing up, walking around the table, and sitting down also controlled the playback of the scene. Standing up would pause the scene and all the characters would become transparent. As a user moved around the table the views of the actors would switch to show which POV the users was nearest.

Once they sat down the scene restarted, after being rewound by a slight amount. Rather than rewinding a set number of seconds, which could have been disjoint or jarring, the scene was rewound to a logical break in the scene, much like a chapter on a DVD. In this way the users were given a moment to get their bearings and it helped them reenter the story as they re-watched some material now from the new POV.

The feedback we gathered from users showed that this use of AR had promise, but the lengthy and frustrating development process taught us that we needed better authoring tools. Since Director did not adequately support AR application development adequately, Moreno was forced to use convoluted and inefficient approaches to create a working system and the process was too complex to support any type of rapid prototyping. It was impossible to iterate on the design or user experience since the code was so complicated. Since we had no opportunity to test the content concepts and user experience early on we discovered technological or artistic shortcomings only after many months had been spent on content post-processing and coding. As a result we had to

rewrite the script and reshoot the video three times. The monolithic nature of the Director application made it *onerous to make small functional changes in hardware* (e.g. using a different type of head tracker or installing the system in a different location).

In Three Angry Men we were exploring an unusual application of AR for the time period; most other application prototypes were military, industrial, educational, or games. Our investigation into the use of the AR medium for dramatic narratives, grounded in media theory, was only possible due to our *close collaboration with a diverse team*. The participation of media experts, design students, and performers was critical to the effective exploration of the medium.

The use of Director was, in part, responsible for making this participation possible by allowing them to engage more closely with the technical side of the project. Emanuel Moreno, an AR neophyte, but a Director expert, was able to lead the development process. The use of a commercial tool made it possible to create a relatively polished and robust application, which we were able to demo to many people over the span of a couple of years. Also, through insights from our collaborators and due to the features of Director we were able to, for the first time, explore the potential of creating AR experiences based on video content, which underscored the need for *an approachable content pipeline and tools that would treat video elements as first-class objects in an AR system*. While this project illustrated the value of leveraging an existing high-level authoring tool it also resulted in frustration due to the lack of flexibility of the resulting software and the inability to iterate on our designs.

## Summary

The impact of this body of work goes beyond the particular artifact or the outcomes of the individual research projects. These experiences led directly to the next stage of this dissertation research, the development of the Designer's Augmented Reality Toolkit authoring tool. In all four of these early projects, working with collaborators, who

were not technologists by training, using a variety of authoring tools and design approaches, provided valuable insight into the needs of non-technologists and their workflow. Key findings from this work included the significance of *approachable access to low level hardware and sensors*, the importance of *a mature content pipeline*, and the need to support *participation by a variety of stakeholders* in the process. These projects also revealed the effect a lack of support for iteration and prototyping has on the quality of AR applications and the power of leveraging existing media development tools in AR authoring. Designing technical systems with creative collaborators, who were focused on artistic exploration and user experience, informed my next several years of work and informed the creation of a unique authoring tool.

# CHAPTER 4

# AUTHORING TOOLS: DEVELOPING DART

Creating early AR experiences in collaboration with non-technologists highlighted the need for authoring tools focused on the workflows of those outside the AR research field. While many of our colleagues at the time were developing authoring tools with specific AR research goals in mind (see Chapter 2), my aims were to address the content pipeline and workflow requirements identified in my earlier collaborative projects. At a high level we were moving from working with "AR as technology" and shifting to "AR as new medium." For this reason we chose to build an authoring tool in Macromedia Director, the Designers Augmented Reality Toolkit (Gandy, MacIntyre, Dow, & Bolter, 2006; MacIntyre, et al., 2004). At the time, Director was a widely used interactive media authoring system with a vast user community. It provided features such as sophisticated media management, which Three Angry Men had proved to us was critical. By leveraging Director, we were able to provide a far more mature and sophisticated system in a variety of practical ways than the typical academic software project. While creating DART, we first included the nuts-and-bolts technical components to support AR (e.g., live video and tracking), but then we designed features that were AR workflow focused (e.g., sketch based prototyping (Presti, Gandy, MacIntyre, & Dow, 2005), capture/playback (Dow, MacIntyre, Gandy, & Bolter, 2004), and Wizard-of-Oz support (Dow et al., 2005b)). The tool was released to the public and was used for a variety of AR applications both at Georgia Tech and elsewhere.

In the following sections, I first present an introduction to the DART architecture. Then I describe the main contribution of this work, which was a was a set of design goals, a four stage development cycle, and the identification of the unique requirements for AR development that together define guidelines for future AR authoring tools. Lastly,

I discuss how we attempted to address these guidelines via a set of design process focused components that we developed over the four years of DART development.

## The DART Architecture

In this section I provide a brief overview of the DART architecture. More detailed information can be found in various publications (Gandy, et al., 2006; MacIntyre, et al., 2004) or on the DART website {, 2006 #226}. Also, Appendices C, D, and E contain overview, release highlights, and programming tip documents that were distributed with DART 3.0.

The DART system is composed of a set of Lingo scripts (i.e., the scripting language of Director) and an Xtra plug-in that extended Macromedia Director to support the development of a variety of AR applications. We chose to develop on top of Director as it provided a very full-featured development environment with an active developer community and cross platform support (i.e., Win and Mac/OSX). By leveraging the features that already existed in Director, including powerful (for the time) 3D and physics engines, we were able to focus our efforts on integrating the necessary AR components with Director's programming model. The Director environment provided programmers with pre-built scripting components in the Lingo scripting language that could be used as-is, modified, or extended by the developer. All the scripts were open and editable, allowing a developer to easily create new components as needed. Figure 5. shows the main DART components and how communication works between them.

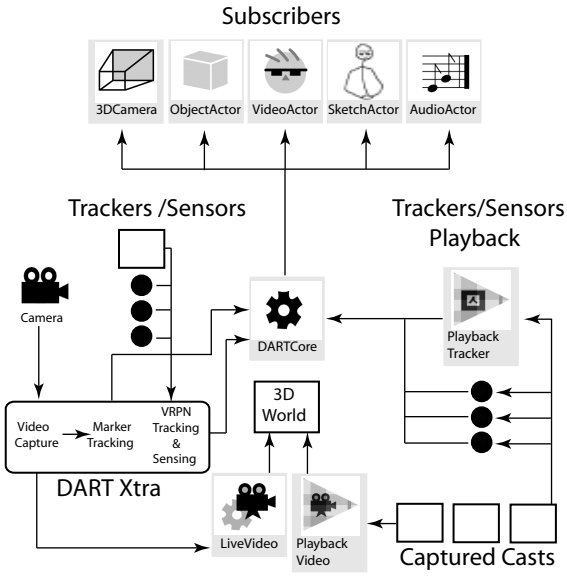



**Figure 5.** The architecture of DART

The Xtra was a plug-in for Director written in C++. This plug-in added low level AR related functionality into Director, including video capture from a variety of cameras, connection to VRPN sources (e.g., trackers, buttons, analogs and distributed shared memory objects), fast video-mixed AR via OpenGL, and marker tracking (i.e., ARToolkit and ARTag ).

The behavior scripts, written in Lingo, were part of the Director authoring environment and could be manipulated just like Director's built-in components. The DART scripts themselves were contained in a series of casts (i.e., the Director structure for organizing content for use in an application). These casts divided up the scripts into logical groups. The DART behaviors encapsulated the high level components that make up an AR application, and provided structured access to the various AR technologies. There were "actors" which represented the content of an application (e.g., 3D models, sounds, HUD elements, lights, etc.) and a "3D camera" that embodied the virtual camera in the 3D world. There were behaviors that connected into the functionality of the Xtra such as "live video," which configured the camera to be captured, for use in video-mixed

AR applications, and “live trackers” which represented trackers, VRPN or marker, in the application. “Transforms” were placed on actors and 3D cameras to control their position, orientation, scale, and to define the parent/child relationships of the scene graph. Transforms could subscribe to trackers, which formed the connection between them and the objects in the application.

Interactivity in DART applications was achieved via a cue/action model. “Cues” were events that were fired when things happened in the application (e.g., the 3D camera reaches a certain position, an audio clip finishes playing, a timer reaches a defined value, a marker appears or disappears, etc.). “Actions” subscribed to cues and waited for them to occur. When the specified cue fired, the action would execute (e.g., start an animation on an object, move an Actor in the 3D world, change the volume on an audio clip, etc.).

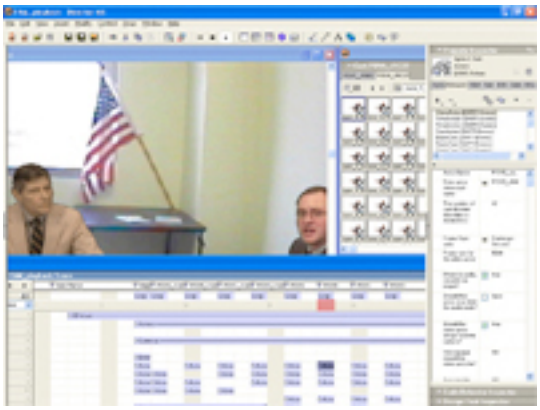


**Figure 6.** An example work session in DART (while debugging the FAM experience). The entire score for FAM is visible, including the nine scenes and most of the actors (each scene is a column in the score). The stage (containing the running experience) is visible, as is part of the content for one video actor, and some of Director’s editing windows.

The development of a DART application progressed, just as any Director application, by placing components on the Director score (see Figure 6. ). To develop a

DART application the author placed scripts either on the 3D world sprite provided with Director or put them in container sprites. When the developer placed a script on the score it brought up a property page, which could configure the behavior. In this manner she could build up the parts of an AR application, placing container sprites on the score and filling them with the desired behavior scripts. For more advanced applications the developer could simply edit code in the behavior scripts, add new scripts of her own, or copy an existing script to serve as a starting point for a new component. DART scripts and custom Lingo could be freely mixed together. DART applications could utilize all the standard Director components and Xtras.

## DART and the AR Design Process

The DART project served multiple purposes. At a practical level it was a useful authoring tool that allowed us to rapidly prototype and deploy AR applications far more quickly and easily than before. The development of the tool over the span of several years, informed by multiple collaborative projects, also helped to reveal the nature of the AR design process. We identified a set of three design goals, which led to a four-stage AR development process, where each stage had associated developer needs informed by our findings from years of experience with AR development, including the four projects discussed in Chapter 3.

The three major design goals for DART:

1. *Support the entire design process*. Creating AR experiences can be difficult. Particularly, the multi-disciplinary design focused explorations that DART was designed to support. The developer must handle the standard software development issues (e.g., performance, modularity, debugging) with the additional challenges of creating 3D content, working with technology such as cameras and trackers, and creating a compelling virtual experience that blends

well with the physical world, both literally and figuratively. In the past, the AR design process had been unsophisticated. The lack of tools and techniques resulted in AR developers spending considerable time creating a single version of an experience that could not be evaluated by users or even by the developers themselves until the content was created, the technology was in place, and the application was almost complete. As a result the design of AR experiences such as TAM suffered since it was not possible to employ the kinds of rapid prototyping and iterative development techniques that are crucial to successful software design in other domains. Ironically, AR is a domain that is even more in need of early evaluation approaches due to the aforementioned complexities that are inherent in AR experiences. An AR design environment must have facilities that allow designers to brainstorm, test their ideas early and often, debug effectively, despite the complex ecosystem of hardware and software that often comprises AR applications, and evolve initial versions of the experience, which may be represented via traditional or low-technology media, into the deployable system.

2. *Provide a powerful, easy to use design environment*. Modern development environments, such as Director, allow applications to be constructed using direct manipulation interfaces, support flexible content pipelines and organization, and support the visual layout of applications to facilitate easy reconfiguration of the experience and the content. Many allow for a layered approach to authoring, where common functionality can be accessed via high level tools such as graphical interfaces, yet those features can be extended via scripting, and, ultimately, the capabilities of the tool can be extended by low level programming. An AR design environment needs to use similar facilities to allow those with varying technical expertise to work with the medium, designers to rapidly replace

content, and to flexibly switch between cameras, sensors and other AR specific components.

3. *Ameliorate the problems of working in the physical world*. Since AR experiences are situated in a particular space, designers are forced to develop and test their applications with the live hardware in the actual location. This limits the amount of testing and experimentation that a designer can do during development. Designers needed to be able to work away from the site of the AR experience, developing and debugging in a modular way by working with simulated or alternate sensors/devices, and to design experiences without having to first deploy the entire sensing infrastructure.

**Stages of the AR Design Process**

The AR design process consists of four stages. It is important to provide the proper authoring support for all of these stages, each of which has different requirements. The following four subsections present these stages and describe the authoring needs for each.

Idea Exploration

Creating AR experiences can be difficult. Being that AR applications are by definition *in situ*, it is important to begin the design process in the location where the experience will be situated. Focusing on ideas, concepts for the virtual content, and user experience without incorporating the context of the space will result in more iterations and wasted work later in the process. Therefore the first step is to gather both functional and aesthetic information to serve as a framework for the brain storming process.

There are a variety of techniques that technical designers utilize to brainstorm, ranging from traditional approaches such as sketching with pencil and paper, to low-tech solutions such as video prototyping (Mackay, 1998), to technology tests like those we

created for the Duran Duran project that are simple working applications. No matter the approach, the goal is to convey ideas to members of the team or users for early evaluation. However a drawback of many of these techniques is that traditionally they can be tedious to create and it is not easy to swap out content or to visualize different scenarios during the process. An AR authoring tool ecosystem must *support these existing workflows and techniques* to allow the team to visualize ideas relatively quickly, while letting them remain technology agnostic in this stage (if appropriate). It is also desirable for those artifacts created in the brainstorming to function as *inputs for the next stages of the process*, rather than being stand-alone assets that lack any connection to the more technology focused steps later in the process.

As we found in our own multi-disciplinary projects, including those discussed in Chapter 3, this step is perhaps more critical with ambitious design driven AR projects due to the variety of expertise and goals of the stakeholders. Communication and collaboration among a diverse team is challenging, yet it is imperative that everyone be able to contribute to early ideation. However, for non-technology focused collaborators to be able to contribute effectively they must have a solid grasp of the *affordances and constraints* of augmented reality. For example, in the Duran Duran project we urged the band members to test the boundaries of marker tracking with technology prototypes so as to set realistic expectations and to inform their brainstorming activities. So authoring ecosystems that either convey this information and/or guide the team towards appropriate design choices are ideal.

Populating the virtual world

As discussed in Chapter 3, one challenge to creating AR experiences is the time and expertise needed to create compelling content (e.g., 3D models, animations, video, sound, etc.). Therefore, an authoring tool must provide a *content pipeline that integrates with existing tools and media types*. In AR, once a team is ready to progress beyond idea

exploration it is also critical that, as they populate the virtual world, they are mindful of this contents *connection to the physical world*.

We have learned from our own experiences with TAM that it can be a mistake to put a large amount of resources into creating content before you have done proper testing or to work disconnected from the physical environment. For both technical and artistic reasons we re-shot the video of the jurors in FAM three times. The artistic missteps were because we had not been able to *evaluate the whole user experience* prior to final content development (Pausch et al., 1996) and, as a result, we had not anticipated the proper way to direct the actors. The technical problems were due to the fact that the students doing the filming had not been able to try out representative video content in the Director application and, therefore, had a misunderstanding of how the cameras should be positioned during shooting to yield proper visual registration. The post-production on the video to create the virtual jurors was extensive, resulting in months of wasted effort. An AR authoring tool should allow for the use of *proxy content* for early experience and technology testing while also supporting a mature content pipeline for high quality content.

Application Development

Once a designer has iterated through early designs and content, the next step is to begin developing the real experience using the sensing hardware, and implementing the necessary application logic that reacts to the sensors. At this stage, an authoring tool must provide an infrastructure that supports *rapid development and modification of applications* as well as application specific extensions to the basic framework. During these tasks the tool must provide the interface and functionality to support *layered application development* in an efficient and robust manner. There was a striking difference in the ease of development for the Beware Home AR installations that relied on C++/OpenGL and the two that utilized Director. The difference was not due to a lack

of technical expertise on the part of the programmers, who were likely more comfortable with C++ than with Lingo, but rather that the Director versions allowed various team members to work at different levels of the complexity depending on their tasks. At the lowest level, I was able to write a plugin using C++ and DirectShow to handle the computer vision component of the systems, while my colleague who was handling application logic could work at the scripting/score level, and the graphic designer and the could easily load and preview content with no programming required. We were also able provide guidance to the team responsible for scripting, performing, and shooting the video content by allowing them to experiment with early prototypes.

The authoring tool must allow for painless *access to a variety of hardware and sensors*. My early projects in Chapter 3 did not provide this easy access, but they showed the value to a project that comes with being able to access a wide variety of tracking, input devices, and output mechanisms. However, this access is inextricably linked to debugging challenges. As our discussions in Chapter 6 will show, one of the biggest challenges faced by non-technology savvy AR developers is the deducing the cause and the solutions for errors that are often the result of problems with AR hardware. My own previous projects have shown that debugging such applications is a different problem than typical software debugging. For example, in GBV, hearing an incorrect audio clip during testing could mean that RF transmitter zones were overlapping, that the state machine logic loaded on the PIC had an error, or that audio files had been loaded on the device in the wrong order. With such a vast error space, debugging becomes overwhelming even for technologists. Therefore, *debugging support* is of critical importance in AR applications due to the range of components and expertise required.

Deployment & Evaluation

It is clear that any software development tool must help the developer create robust and reliable applications, however, this is more challenging to achieve with AR

authoring tools. As will be discussed in Chapter 6, interviews with AR developers highlighted the struggles associated with exhibiting and maintaining an AR installation. Some of these struggles can be eased by flexible hardware access, which allows for *replacing or substituting new sensors* as needed (e.g., the system can be developed with one tracking solution and then another can be used for deployment). However, depending on the sophistication of the system, there are a variety of components that must be monitored during deployment (e.g. batteries will die, sensors must be recalibrated, HMDs/monitors go into sleep mode, etc.). Our experience with projects such as Duran Duran and the BewareHome provided insight into the manpower, technical expertise, and stable computing platforms that are required to keep an AR experience running "in the wild." An AR authoring tool should generate stable responsive applications, but there may also be a need for tools that support debugging and *monitoring during deployment* to help the researchers/crew/staff maintain consistent quality.

The other aspect of this stage is the need to *test and evaluate AR applications*, an area that traditionally has been underexplored in AR authoring tools. In the four early AR applications of Chapter 3, we missed opportunities to gather a deeper understanding of the AR medium. At the time, the challenges of simply creating a working AR application were so great that we did not have the resources to do formal evaluations. For AR to mature as a field it is vital that authoring tools provide facilities for user evaluation and debugging later in the process. To that end, authoring tools should provide methods of *storing and analyzing application data* gathered at run-time, a relatively simple feature that would have facilitated more rigorous evaluation of our early projects. Our experiences also highlighted the need to test work-in-progress systems, manually simulate sensor data, contextual information, or system intelligence before fully implementing them. Therefore, there is value to AR authoring tools *supporting standard HCI evaluation techniques* such as Wizard-of-Oz.

## Addressing the Needs of AR Authoring

While the use of Director helped us to provide many of the non-AR specific requirements of a modern media development tool from the start, our research was concentrated on identifying the unique challenges of AR authoring and designing DART tools that addressed them. The challenges include an authors need to: test applications in situ, switch between tracking solutions depending on the stage of the design process, prototype the entire user experience, and overcome the difficulty of capturing application design ideas and communicating them to others. These tools had to not only meet AR design requirements described in the previous section, but they also had to function in a way that was useful to non-technologist and that fit into their workflow. The following sections describe the DART specific features that support each stage of the AR design process.

### Capture/Playback (*Exploring Ideas*)

The capture/playback behaviors let the designer begin the design process during the *exploring ideas* stage by gathering data about the location where the experience would be situated. DART allowed the designer to easily capture video as well as sensor and tracker data in a space (Dow, et al., 2004). This capture facility could be integrated into a DART application and then data could be replayed in any DART application in place of live data.

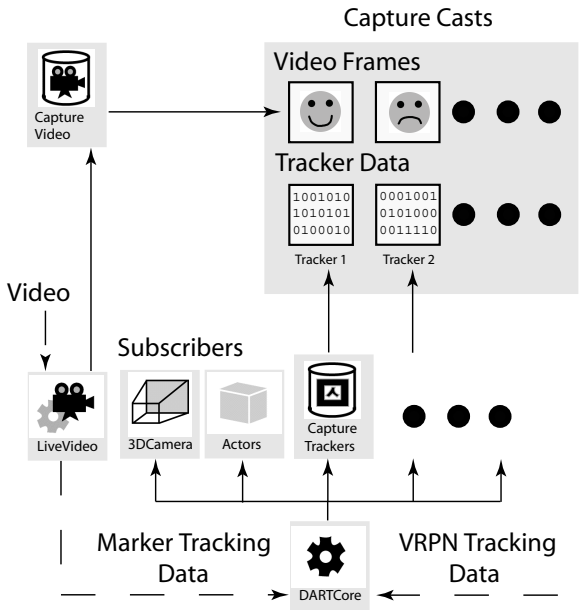



**Figure 7.** Architecture of a DART application that is capturing data

The architecture let the author easily swap in and out live feeds and pre-recorded data. Figure 7. shows how the captured data is collected as part of a standard DART application. Any DART application could capture data by including "capture video" and "capture tracker" behaviors on the score. The resulting captured application could be a fully functional experience or it could be a bare bones application that simply recorded data. Captured data could then be replayed in any DART application via the "playback tracker" and "playback video" behaviors. Since the designer could control the abstract clocks that drive DART applications, the entire experience including captured data could be paused, rewound, fast-forwarded, and played at arbitrary speeds at run-time.

All of the captured data was stored into cast libraries, where it could be viewed and modified if necessary. During playback, a designer could choose to replay a certain portion of a captured data set or they might mix/match data from several independent captures. When sensor data was played back, it was essentially fed back into the system as if it were a live sensor allowing for swift replacement of live data with prerecorded data and vice versa.

In the DART 3.0 release, we included a feature called ReplayAR that provided the ability to control capture and playback in a way similar to modern digital video recorders. An application could start with live video and trackers active. At any time the developer could pause the live feed and rewind into the saved buffer of video and tracking data. This allowed the developer to run an application live, but at any time back up and replay to investigate anomalous behavior or to revisit a moment of interest. Onscreen controls allowed the developer to pause, rewind, play, and fast-forward. They could hit a hot key at any time to save the data buffer to disk. The saved buffer could then treated the same by the system as any other captured data

The capture/playback facilities let an author work with real tracker data and video from a target location wherever and whenever they wanted for early design activities as well as debugging and testing. However, experience revealed that these features could be utilized even in live applications. For example, we found that the recording tools could be used to quickly create animations (e.g., an author defines an animation for a character by moving a tracker in the desired motion and then uses that data to drive the movement of a 3D model).

**Sketch Annotations and Video Prototyping (*Exploring Ideas*)**

Sketched content is commonly used in pre-visualization to create animatics (i.e., animated storyboards, used during film and television pre-visualization). Not only has sketched content been shown to be useful for rapid content creation, the use of sketches can enhance the design process by tacitly freeing people to suggest radical changes (Landay & Myers, 2001) and can convey more of the designer's intent than quickly created 3D content. As a complement to the capture/playback infrastructure we created a DART component that allowed for the creation and playback of sketch annotations overlaid on the scene (Presti, et al., 2005). Inspired by Pixar's Review Sketch (Wolff,

2004), this interface not only allowed the AR designer to “draw” on top of the “real world” video, but also to generate 3D billboarded planes from the 2D sketches.

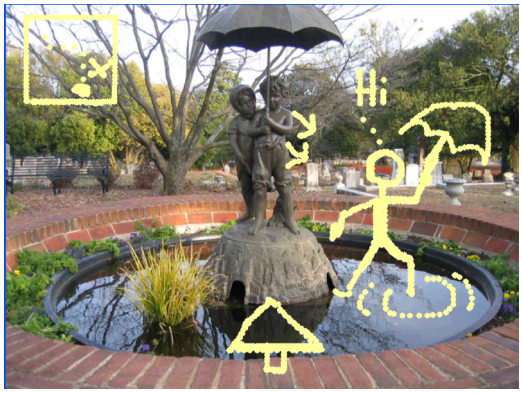



**Figure 8.** A DART sketch prototype of an outdoor AR application

This feature supported the author in quickly getting initial ideas into a concrete form. When the author wanted to add a sketch on top of the scene she paused the playback in the application and drew whatever she wished on the screen. The next time the application was run the sketches were replayed on top of the captured video automatically (see Figure 8. ). The 2D sketch annotations could also be transformed into 3D content “sketch actors” (to be discussed in the next section) for use in the next stage of design that incorporated tracking data.

Video prototypes could also be created using regular DART actors and time-based cues. The designer would simply put 3D object actors, sketch actors, audio actors etc. on the score and place the transform scripts to define their 3D positions and orientations. Once the designer had placed the actors, she could utilize time cues to bypass the need for application logic. The time cues simply fired an event at a specified time; actions placed on the actors would receive them and respond accordingly. In this way the designer could define the types of actions that might occur in the final experience

without having to create the possibly complex logic to generate them from sensor input. For example, the designer might create a captured data set where at 5 seconds in the user walks through the door and at 6.3 seconds she makes a gesture with her hand. The designer could then define time cues called “door_enter” and “wave_hand” that fire at those respective times. The actors would then respond to those cues in the desired way (e.g. an action “shows” an object while another action triggers a “start animation” on the same actor). Although these time cues would not be useful in the live application, the actions would be, and in the meantime the designer could explore various approaches rapidly even if complex algorithms or technology would be required to make it “real.” Once the user experience had been fleshed out the designer could focus on how to add the necessary intelligence to the system. The result is a video prototype that does not require wasted effort, can be rapidly modified, and that can evolve into a deployable version at the end of the cycle.

**Sketch Actors (*Populating the Virtual World*)**

While exploring the AR design process and the challenges of initially populating the virtual world, we were influenced by the use of animatics and storyboards in the film industry. Using this approach in pre-production, directors are able to visualize their shots in a cheap and easy manner. Therefore, DART supports the concept of “sketch actors,” or flipbook style animations created from a set of still images. The designer would create a set of sketches illustrating what an object in the scene will look like (e.g., these could be drawings done by the design team, or still images such as photographs) and define the timing that determines when each image will be displayed. The sketch actor was placed in 3D space in the world just as any other actor and supported the common set of actor functionality (e.g., start, stop, hide, show, etc.). This allowed the author to populate the

world with rough content for early prototyping and user experience testing while avoiding the need to devote resources to content development (see Figure 9. ).

(a) (b)

**Figure 9.** (a) A DART prototype of FAM using sketch actors. (b) The final FAM application

**Proxy Content (*Populating the Virtual World*)**

DART supported the concept of virtual and physical objects, so that actors could represent either virtual content to be registered visually with the physical world or they could represent elements of the physical world that would be used for occlusion and physics (see Figure 10. ). This feature, coupled with the direct manipulation method of application development on the Director score, resulted in a suite of techniques that allowed a DART designer to quickly modify or swap content for debugging and evaluation. For example, when we would begin placing 3D virtual objects in the physical world for a new AR experience, we would often use proxy content, such as simple primitive objects or sketch actors, eliminating the problems that might arise with more complex models and allowing us to begin evaluation and testing before time was spent on final content. This proxy content was easily replaced later in the process.


(a) (b)

**Figure 10.** (a) A virtual dog object is visually occluded by the physical cup (b) a virtual ball bounces off a physical book. This interaction is possible because both objects are represented as physics enabled "actors" in the DART application

Meanwhile, in this stage of the design, a designer will often start mapping the extent of the space, determining where virtual content should be placed, and verifying that the tracking is working. Proxy content that might not appear in the final experience can support these activities and serve as spatial markers to the designer (e.g., indicating the location of the origin of the world coordinate system or showing where a physical object will be placed in the world). For example, in Four Angry Men there were position cues defined that would fire when the user entered a zone around each juror. In the design stage we used semi transparent spheres to visualize these zones to help us determine their optimum size and to debug the behavior of the position cue (i.e., we could see when the user entered the zone and could verify the cue fired appropriately). It was very easy to later swap out these stand-in actors for others, or, in the case of objects used for debugging, move them to an inactive portion of the score or make them temporarily invisible, keeping them available for later use.

**Tracking Architecture (*Developing the Application*)**

An important element of any AR application is tracking. One obstacle that prevents designers from creating AR applications is the expertise and domain knowledge required to understand the tracking/sensing technologies and the complexities intrinsic to working with the hardware. These technologies are often difficult to work with for a variety of reasons. They can be expensive and difficult to configure. Interfacing between the sensors and the computer application can be complicated and require low level programming expertise. The reliance on real-time sensors means that it is difficult to develop the application off-line without the hardware available, and it can be onerous to change which type of tracking technology the application uses. In DART we implemented an approach to tracker management and leveraged existing technology such as VRPN to alleviate many of these problems (Gandy, MacIntyre, & Dow, 2004). The result was a flexible approach to tracking that allowed for experimentation, off-line development via capture/playback, prototyping (e.g., sketch support), wizard-of-oz testing, and easy migration from one tracking technology to another (e.g., switching from a 6DOF tracker to marker-based tracking) (see Figure 11. ).

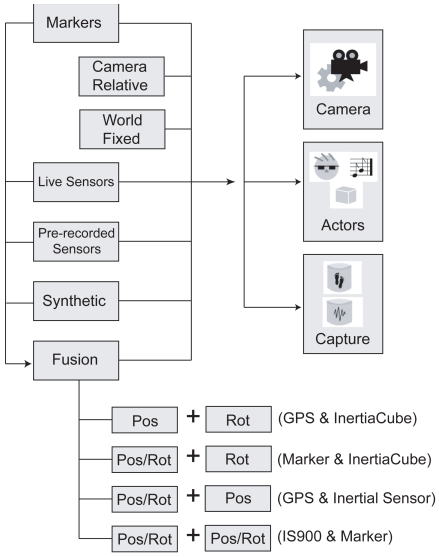



**Figure 11.** The tracking architecture of DART

The DART tracking infrastructure also made it easy to create various kinds of synthetic trackers because the tracking data was routed through a central location, all consumers of the tracking data used a common subscription model, and all tracking reports used uniform timestamps/formats. For example, a designer could fuse the data from two different trackers to create a new "tracker" by combining GPS position and inertial orientation data.

**Layered Authoring (*Developing the Application*)**

It is not possible to proactively provide all the functionality any designer would ever need for any AR experience with a development tool such as DART. Design tools that do not have facilities for creating custom code, where the applications must be designed entirely through manipulation of the provided components, will result in the development of a narrow niche of applications. We found that for any application of sufficient size or complexity it was necessary for the designer to develop custom additions to DART. Fortunately, when developing real applications, a benefit of DART was this ability to develop at many levels. At the highest level the designer could simply use the components provided with DART and design the application visually. At the next level the athor could take a standard component, copy it, and modify it to work differently for her app. The developer could also write custom Lingo code, hooking into the DART infrastructure (e.g., actors, tracker subscription, cue/action architecture, etc.), leveraging DART in the most efficient manner for her application. Lastly, a developer could implement custom Xtras written in C/C++, adding support to the Director environment for new services. By providing this multi-layered authoring support DART did not limit developers in the type or sophistication of the applications they created and allowed them to work at whatever level of abstraction they found comfortable.

**Wizard of Oz & Visualization Tools (*Evaluation and Deployment*)**

The Wizard-of-Oz (WOz) simulation method is a prototyping approach widely used among researchers and professionals in HCI. A "wizard" operator generally plays some role in a work-in-progress computer system, manually simulating sensor data, contextual information, or system intelligence. The Wizard-of-Oz functionality in DART could be used both during the design process and during deployment, as a way of overcoming the shortcomings of technology. It was made possible by the flexible tracker architecture and use of VRPN.

The WOz tools in DART leveraged a broadcast/subscription architecture (Dow, et al., 2005b). Two scripts that were utilized in both the wizard and "puppet" applications would establish the networking connection and enable the wizard interface to trigger any actions available in the user application. By using common naming conventions, events could be triggered locally or by a remote wizard. The wizard could also spoof any of the tracker data streams. DART could automatically generate an interface based on the events available in the puppet application, lowering the barrier for using WOz prototyping as an evaluation strategy.

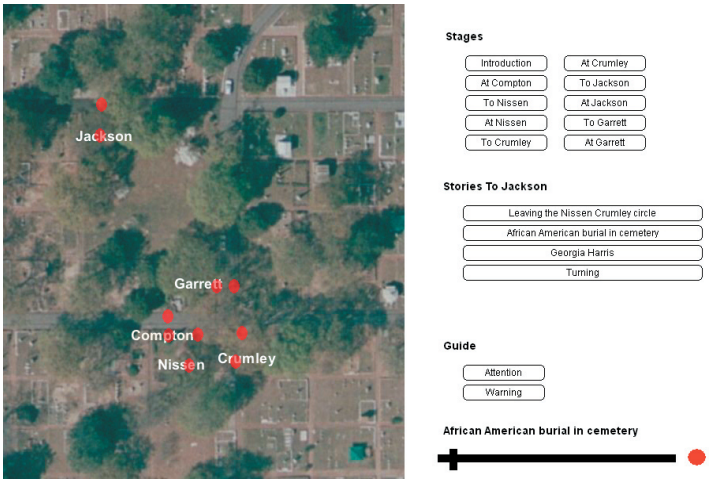



**Figure 12.** A WOz interface for the Voices of Oakland experience

The wizard interface could also be customized to control the puppet using a mix of built-in Director widgets and DART behaviors. For example, in VOO the designer placed an overhead map image in the WOz application and attached a "map tracking" behavior that generated synthetic GPS tracker reports that appeared to the puppet application to be real GPS reports (see Figure 12. ). Since both interfaces were developed in DART, we were able to integrate part of the user interface of the puppet into the wizard interface so that the wizard operator would experience the same application state as the user. In VOO we used this strategy to allow the wizard to listen to the same audio segments, follow along with the user, and decide what content to display (Dow et al., 2005a).

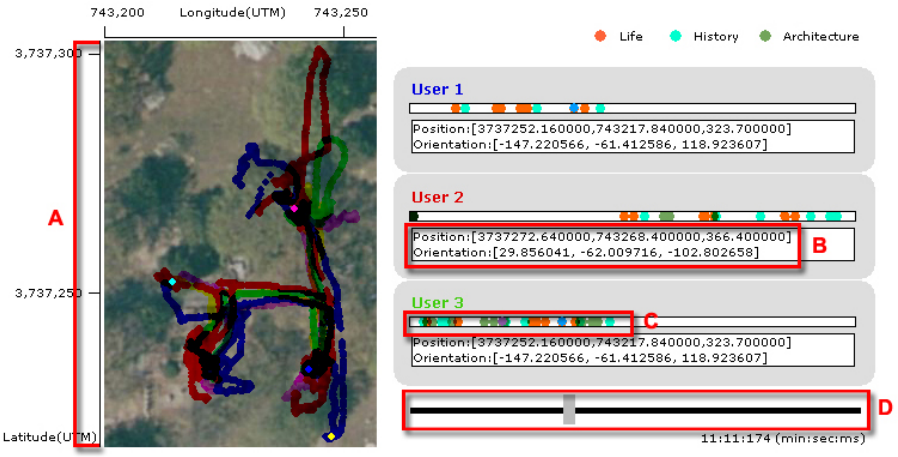



**Figure 13.** Visualization of VOO created with DART behaviors (*DataGraphs, Observers, TimeSlider*) inside Director. a) GPS data for 5 participants with dynamic circles showing the user's position at a particular time. b) Textual representations of GPS location and head rotation data. c) Graph of button interaction over time. d) Slider for control of DART's abstract time.

To help designers evaluate an experience, DART also included tools for visualizing captured data textually and graphically. By having the tools integrated with DART, designers had the ability to visualize live data in parallel with previously

collected data, enabling real-time analysis of user performance. Visualization tools in DART could show both static, cumulative data as well as a single data value at any particular time on the abstract clock. For example, in VOO we visualized the GPS data for each participant on the same image to get an overview of user movement (see Figure 13. ).

## Summary

While DART was not a panacea for all of the problems encountered when working with AR, or any media that mixes physical and virtual worlds), it took a significant step toward enabling designers to work with this new medium. We focused on meeting the overall goals of supporting the entire design process, providing a powerful and easy to use environment, and ameliorating the problems of working in the physical world, while attempting to satisfy the tool requirements we identified for each of the four stages of AR development. Throughout the AR design process there are authoring needs unique to the medium, the need to: *design and test in situ*, allow for *seamless switching of sensor/tracking solutions*, support the *capture and communication of early design ideas*, and *prototype the entire user experience* (virtual and physical). These findings yield a set of guidelines that can inform future AR authoring tools:

1. *Rapid prototyping support is critical*. The tool must be built with the philosophy that designing and iterating on the physical components of the experience are as important as the virtual.
2. *Early ideation artifacts should transform into technology prototypes.* When possible a tool should assist in the use of low or no technology content as inputs to early versions of the AR system.
3. *Effective communication of ideas between stakeholders should be supported.* This encourages the creation of sophisticated AR applications that result from the

collaborations of diverse teams. The authoring ecosystem must allow contributors with varying expertise and goals to participate in the design and development process

   a. *A layered development environment* enables these collaborators to engage with the AR application directly at an appropriate level.

4. *A mature content pipeline is required.* Such a pipeline not only results in high quality final versions of the content, but also encourages experimentation and iteration throughout the process. An authoring tool should make it quick and easy to import content, while recognizing the need to flexibly substitute versions of content for comparison and testing.

   a. *The concept of proxy content* in a tool allows the designer to evolve content throughout the design process informed by evaluation

5. *Access to trackers and sensor hardware* assists the designer in exploring the full potential of the AR medium. Ideally, an authoring tool should let developers work with a diverse set of arbitrary hardware without extensive domain knowledge.

   a. *The ability to easily substitute or replace technologies* aids the designer in during brainstorming, debugging, and deployment

6. *Debugging support should assist in the analysis of the entire AR system*. AR applications often involve a chain of hardware and software components. The biggest challenge in debugging can be determining what element in this chain is causing unexpected behaviour. Authoring tools should assist the designer in identifying the source of a problem as well as guiding them toward a resolution.

7. *Supporting deployment includes monitoring and debugging functionality.* It can be challenging to keep an AR experience operating for significant periods of time. An authoring tool environment should allow team members to maintain an awareness of system state and to scrutinize more closely when problems occur.

8. *Formal evaluation is critical to the future of AR.* Authoring tools should support designers in the logging and visualizing of user data. The tool can also provide features informed by HCI evaluation techniques that lower the barrier of entry for undertaking formal studies of AR experiences.

In Chapter 5 I present internal projects built with DART and in Chapter 6 I provide in-depth reflections from external DART users. These chapters offer insight into the validity of these guidelines, the use of DART features in reality, and how DART did and did not meet the requirements of an ideal AR authoring tool for non-technologists.

# CHAPTER 5

# BUILDING WITH DART

The validation of our AR design process guidelines and the evaluation of DART comes from an examination of the variety of experiences they were used to create (Edwards, Bellotti, Dey, & Newman, 2003). In this chapter I will highlight a set of applications that illustrate the variety of collaborators, the diversity of domains, and the real-world workflows that DART enabled.

## Four Angry Men

The follow-on to Three Angry Men, Four Angry Men (FAM) was the first full application built in DART (Gandy, et al., 2006). This process of building a more sophisticated version of a previous application was one method of validating DART and its concepts. Many of DART's features were based on needs we encountered while designing and implementing TAM. We chose to focus on supporting a single application type, initially, so as to validate DART for one specific domain before adding additional features, as recommended by Edwards et al. (Edwards, et al., 2003). As a result, DART has sophisticated facilities for dealing with *video based content,* which was unusual in an AR tool, but had proven ideal in TAM. This feature subsequently evolved into to several sketch-based *rapid prototyping* features.

To create FAM, we rewrote the TAM script, adding another juror character, and reshot the video (see Figure 14. ). The use of DART allowed us test the experience with *proxy content* prior to the video shoot. This test proactively highlighted an error in how we planned to shoot the video, which we fixed before shooting. We also sketched a set of very rough storyboards, consisting of stick figures, illustrating the types of movements and jurors' reactions we wanted, and created sketch actors from them (see Figure 9. ). These actors were placed in the 3D world so that they would appear to be sitting at the

physical table, just as the final content would. We quickly recorded audio of the script using with our own voices. This allowed us to *debug content-oriented issues* before too much effort was put into creating the real videos and audio. Once post-production was complete on the final FAM video content we simply replaced the sketch actors in the application with video actors. The post-production of the video was time consuming, but was less than with TAM due to the specific video actor support provided by DART. The implementation of the final experience took only a couple of weeks, versus six months for the original TAM application.

**Figure 14.** The four virtual jurors of FAM.

Lastly, once the experience was implemented, due to the *flexible tracking architecture*, we were able to easily stage FAM in a variety of venues and tracking conditions, including a portable version that presented a miniature variation of the experience on a fiducial and a presentation version that used captured data.

### AR Karaoke

FAM inspired AR Karaoke. It was a prototype that provided a "karaoke for acting" experience, a way for users to experience acting out their favorite movie scenes with virtual actors (Gandy et al., 2005). I led a student team who developed the project as part of the AR Design course.


**Figure 15.** The user's view of the AR Karaoke "The Princess Bride" application. Icons prompt the user in the sword fight. Text at the bottom shows the lines the user should be speaking. The progress meter in the top left shows how long the user has to deliver the line. The map view in the top right provides blocking guidance.

ARK was an example of the type of *quick application exploration* that was not possible prior to DART; the whole system was built in a few weeks. As there was no time to create polished content, a working version of the system was developed with *sketch actors*, both hand drawn and still photos. We were able to create three different experiences using the architecture: A scene from "The Wizard of Oz," a famous segment

from “The Princess Bride” (see Figure 15. ) that explored the interface required to lead a participant through a physical scene (i.e. a sword fight), and a new version of FAM. This FAM experience was created to assess the value of user interaction during the experience. Little development was required due to the prototyping features of DART that allowed me to quickly add ARK components to the existing FAM application. This is an example of how DART allowed us to quickly explore more *“risky” ideas with little resource investment*.

### Voices of Oakland

The Voices of Oakland revisited audio-only AR concepts that I had first explored with GBV. The experience led visitors around the Historic Oakland Cemetery in Atlanta, providing a location-based audio tour of the graves of several historically interesting people (Dow, et al., 2005a) (see Figure 16. ). The project was a *collaboration with a diverse team* including Dr. Jay Bolter and students in the Digital Media program as well as the curators of Oakland Cemetery. The site was also used for a variety of prototype DART-based AR projects as part of the AR Design course.

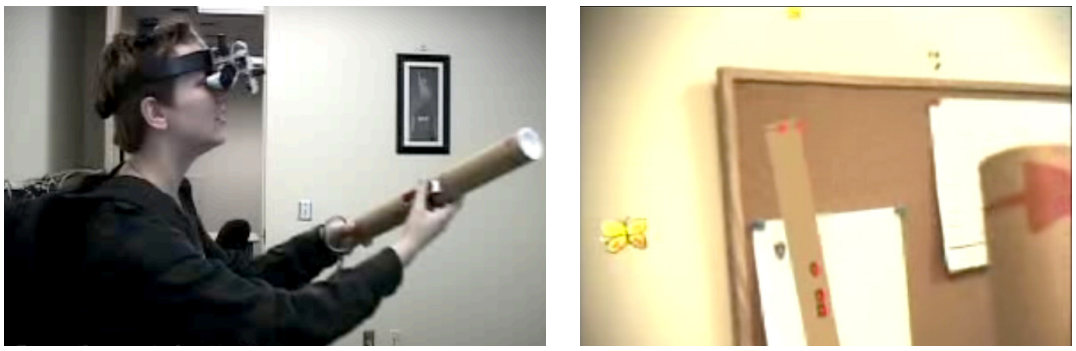
In the case of VOO, and in several other AR Design student projects situated in Oakland Cemetery, we began the design process by *capturing video, GPS, and inertial sensor data* in the cemetery. The designers simply walked through the space interacting in ways we thought would be relevant for the final experience (e.g., walking up to a certain head stone, looking a famous sculpture etc.) Once the data was captured it could be used in a variety of ways in different stages of the design process.

**Figure 16.** A visitor experiencing the Voices of Oakland

During implementation we found that GPS error made the experience unpredictable and confusing for users. As a result we implemented the final application using a WOz interface such that a "wizard" could manually control the location of users to achieve an acceptable level of tracking resolution. The system was evaluated with real users and provided findings about users usage behaviors and preferences in an audio based tour. This system *motivated and informed the development of the WOz components*.

### Butterfly Effect

The Butterfly Effect game was an attempt to create an indoor AR game that did not rely on extensive knowledge of the geometry of the environment (Norton & MacIntyre, 2005). The player was tasked with catching all the virtual butterflies in a space. She could move the butterflies into reach by performing rotations of the butterfly swarm around a "tornado stick" controller (see Figure 17. ).



(a) (b)

**Figure 17.** The Butterfly Effect game a) A user controls the game via the phidget-based "tornado stick" (b) A screenshot with the virtual butterflies, the virtual representation of the tornado stick, and the physical stick.

A Master's student developed the game in a few weeks. The game utilized another feature of DART, which was support for the *phidgets hardware platform* (Greenberg & Fitchett, 2001). Phidget support allowed DART authors to prototype *custom input and output hardware* and easily use them in live AR applications.

### AR Design Class

In the spring of 2004, DART was used in the AR Design course. This class of undergraduate and graduate computing and digital media students utilized DART to create a variety of AR prototype experiences. They made heavy use of the *capture/playback* (see Figure 18. ) and *sketching* capabilities, which allowed them to test and demonstrate ideas in the absence of trackers, before resources were sunk into development. They were able to use these features in a workflow that started with, what was in essence, a mock-up video, made in DART, that could easily be evolved into a working application throughout the semester.

**Figure 18.** A student in the class captures data at Oakland Cemetery

This *video prototyping* technique emerged naturally from students in our AR Design class. We intended for the students to use the captured data to create applications that would work in a live experience, but noticed that instead of programming complex behaviors early in the design process, the students were utilizing the time cues to make a prototype that looked correct with the captured data set. We realized this was a useful approach that allowed them to realize their ideas quickly and to iterate through several approaches. The creation of the application logic is more appropriately left for the next step in the design process.

### Pachinko

In 2005 we were asked by tracker manufacturer Intersense to create a demo, using their new Viztracker IS1200 product, in DART. This is an example of using DART's facilities to *prototype and quickly deploy* an AR application. The application was based around a large banner of printed markers, utilized by their custom camera hardware for tracking. The user would hold up a tablet, "magic window style", and tilt the device to guide a ball through a virtual pachinko style board that populated the poster.

**Figure 19.** A SIGGRAPH 2005 visitor interacts with the Pachinko game

Our *flexible tracking architecture* made it easy for us to use their beta tracking product and to develop an application ready to be shown to the public in only a couple of weeks. This system was demonstrated by Intersense at SIGGRAPH (see Figure 19. ) and ISMAR '05.

### DART the Dog

The "DART the Dog" project was a *close collaboration* with a digital media Ph.D. student. He was a 3D artist and media theorist who was also capable of delving into code. Although DART was still not that accessible to non-technologists without any programming ability, it was well suited to this type of designer and project. The *mature content pipeline* provided by Director made it possible to work with high quality models and animations created in 3D Studio Max. And it was the content that was critical to the success of this project. The behaviors were simple; DART the Dog was a virtual pet that would run around on a marker, responding to cards that represented various items such as a food bowl and a rubber ball, and interacting with them (see Figure 20. ). This is an example of a simple but compelling content-centric application that was well supported by DART. Previously in AR development, with the process being so lengthy, we might have shied away from creating a "toy" application of this type, as it would not have

seemed worth the months of development time. However, DART gave us the *freedom to explore* seemingly trivial or fun applications throughout the AR design space.


**Figure 20.** DART the Dog occluded by a physical coffee mug

To create a compelling experience it was necessary for me to work closely with the designer to *convey the unique content requirements of AR*. Much of the design process involved creating sample dog models and animations and then evaluating them in the AR environment. Often, a model that looks good in the modeling tool is inappropriate when placed into the AR setting. For example, in this type of marker-based application the user is mainly looking at the content from an isometric-style view from above. Early versions of the dog had him looking at the ground too much and as a result the viewer would miss many of the cute facial animations. Also, unlike a typical 3D game world, an AR user can look anywhere at anytime. Therefore, all parts of the model must be complete and compelling (e.g., you cannot assume the user will always be looking at the dog head-on). DART supported this *rapid iteration* and provided the designer with *instant feedback on his content*.

We were surprised at the positive response from users of this prototype, especially children. Anecdotally, we demoed this experience to a wide range of people at a variety of exhibits and demo showcases. The main feature that resonated with visitors, particularly since AR was very new to them, was simply having a virtual character in the world with them and having the ability to perform *tangible interactions* with it (e.g., moving the food bowl while the dog was eating and having him chase after it). Children seemed drawn to interact physically with the character; we observed multiple instances where children wanted to see themselves in place with the dog (e.g. taking pictures with the dog so that it appeared to be in their mouths). This experience illustrated the importance of *polished and appropriate AR content* to the value of the user experience.

## TUI Toolkit

We first approached the idea of utilizing DART for rapidly prototyping and developing tangible user interfaces in response to a workshop on TUI toolkits at Pervasive 2004 (Dow, et al., 2004). The focus of the workshop was on the need for toolkits that supported exploration of interactions in the physical world. Our presentation on the possibilities of DART in this domain helped us realize that there was yet another community that could leverage our tool.

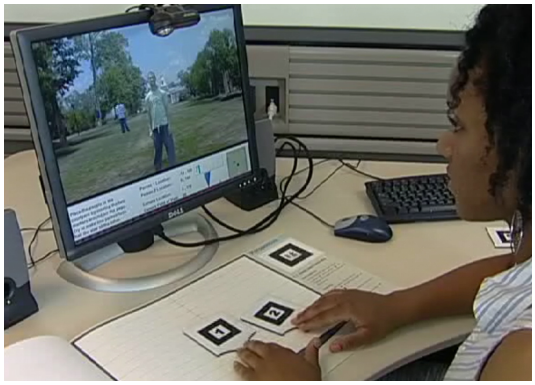


**Figure 21.** A student interacts with the TUI application.

The opportunity to extend DART's features to support TUIs came in the form of a collaborative project we carried out with Morehouse College faculty and students. They had an National Science Foundation funded project to create novel technology experiences that would excite K-12 students about STEM topics. The project was focused on using fine art experiences to subtly present science topics such as optics and fractals in an approachable way. Over time the team decided to explore TUI interfaces; students would interact with a printed book and tangible paddles to explore painting and photographs. The team was eager to use DART for authoring and did not have the time or budget to craft a custom hardware and software TUI platform. This prompted me to begin a sub-project designing and implementing DART components that supported rapid development of cheap and easy marker-based TUIs (Gandy, Jones, Robertson, O'Quinn, & Johnson, 2009). During development our team *collaborated with education researchers, scientists, and artists* to craft the content and experiences (see Figure 21. ). Morehouse undergraduates assisted in the DART development. The *layered authoring environment* encouraged all of these collaborators to directly participate in the development process. The *extensible architecture* of DART allowed us to add these new features relatively quickly and easily, similar to our experiencing adding Phidget support. The use of Director and the integration of components specific to TUI development supported this collaborative design process.

## Summary

While there were dozens of internal projects built with DART, this chapter highlighted projects I participated in that exercised key features of DART.

These projects highlighted the value of various aspects of DART including:

- *Layered authoring* was a powerful feature of DART. It allowed us to develop applications quickly and enabled us to work with collaborators with a diverse set of skills
- DART permitted us to work with *early ideation content* such as sketches, storyboards, and photographs, which could later became the basis of early technology prototypes
- The *content pipeline* provided by Director and DART not only resulted in polished final content, but allowed us to iteratively explore content approaches as we learned how to fully leverage the affordances of AR.
    - It was valuable to have extensive support for AR *video content*.
    - *Proxy content* was leveraged in a variety of ways for laying out the virtual and physical scenes, prototyping, debugging, and evaluating the user experience
- The *flexible tracking architecture* allowed us to develop systems away from the actual site or intended tracker, debug easily, quickly port experiences to new environments and conditions, and even extend the tool for new application domains.
- The *capture/playback* functionality not only improved the debugging process, but we found it could be used to create very early "video prototypes", as well as content and control data for the final application.
- The *WoZ architecture* proved useful both for early testing of experiences as well as a component of a deployed system.
- The *hardware access* afforded by DART allowed us to work with a wide array of technologies including "home brew" controllers built with phidgets to beta versions of new tracking systems.

These projects also illustrated the value and challenges of collaborations with diverse teams and the overall power and flexibility of DART. Our experiences with these projects demonstrate the validity of our authoring tool goals, development stages, and guidelines presented in Chapter 4.

# CHAPTER 6

# REFLECTIONS ON DART

While reflecting on projects done internally is valuable, the most significant feedback regarding DART came from our community of external developers. Fortunately, we have a large user community to mine for data. DART has been available since 2003. There were three major releases of the toolkit. In that time we have recorded 11,573 downloads by 4,280 unique users (as of November 2011). Although the last release of Director was in 2008, people are still downloading, and possibly actively using DART. Even in 2011 there were 377 downloads and we still receive the occasional support request on our developer mailing list. We have anecdotal evidence that DART was used by a significant number of people for a large number of successful projects. Our mailing list activity shows hundreds of posts with developers asking for help and requesting features. Web and YouTube searches reveal DART projects that we never knew existed. We have also had many in-person conversations at conferences and exhibitions where external users have approached us to discuss DART and to thank us for creating it.

However, these anecdotes do not provide the validation and deep reflections on DART that could inform the design of future authoring tools. In "Stuck in the Middle", Edwards et al. discuss the challenges of evaluating infrastructure software (Edwards, et al., 2003). The evaluation of end-user applications is a well-explored field, but different approaches are required when measuring the value of toolkits. While a toolkit can be assessed by its technical capabilities, ultimately, what matters is the value to the end-users. They describe the evaluation of toolkits indirectly through the context of use and through applications built with the tool. Other toolkits have been evaluated by studying their use by target developers. Some were used in the classroom setting and the resulting

student projects were evaluated (Dey, et al., 2001; Greenberg & Fitchett, 2001; Klemmer, et al., 2004). Others, such as Designer's Outpost and Denim, were used by their target professionals (e.g., web developers) (Klemmer, et al., 2001; Lin, Hong, & Landay, 2001). Some were released, like DART, to the wider community (Dey, et al., 2001; Greenberg & Fitchett, 2001; Lin, et al., 2001).

Therefore, the goal of this chapter is to present the findings from in-depth interviews (see Appendix B) with eight external DART developers. They worked with the tool from our alpha version in 2003 till present day. The majority of their projects were developed from 2004-2008. I also interviewed three AR experts that did not use DART, but who have a depth of experience that provides perspectives on AR authoring. Overall these participants represent the full range of AR creators from those focused heavily on technology and computer science contributions to artists using AR as one of many tools in their exploration of media (see Appendix H).

In this chapter I will introduce the eight developers and the projects they created with DART. Then I will discuss their feedback regarding the positive and negative aspects of authoring with DART. This feedback not only provides data on the utility of DART but also reveals general findings about the unique requirements of AR authoring. I will also interweave comments from the three AR experts regarding the challenges of AR authoring. Lastly, I will conclude with a discussion of how this feedback informs the current requirements are for modern AR authoring tools.

## The Developers

While I interviewed eight developers, in this section I have organized their feedback by group/project. The result is four developer groups that represent the full spectrum of DART users, from the highly technical to those with purely artistic goals. In the following section I present each developer/group persona and describe their project(s) with DART, discussing why they chose DART, highlighting their interesting uses of

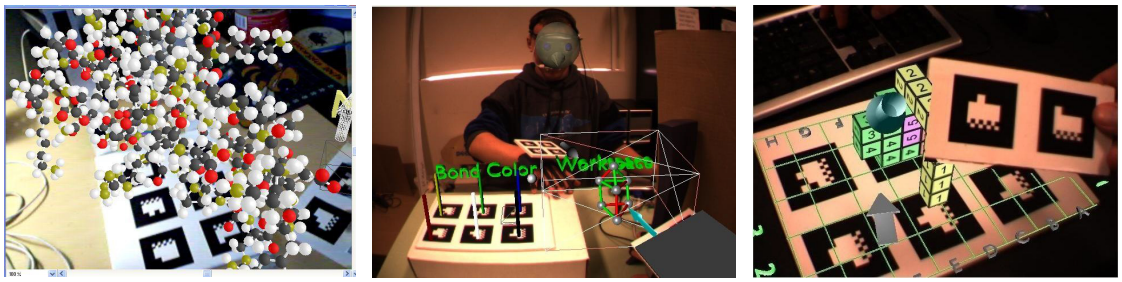
DART, summarizing their positive and negative feedback, and concluding with their feedback regarding how DART (or AR authoring tools in general) could be improved.

### The Computer Scientist

This developer (who will be referred to as TECH) is the example of a technologist and *a power user of DART*. A computer scientist, he used DART to drive a series of HCI experiments and is an example of a developer who not only used, but *extended DART* at a low level. His projects included several AR applications developed over a period of three years. In the beginning, he used DART to create an AR version of an existing VR application. This first prototype revealed fertile ground in the area of AR collaboration and led to one pilot and two full experiments. This was the first example of utilizing DART to support *evaluation and experimentation* as well as implementation.

#### The Projects

Beginning in 2005, TECH utilized DART for a series of experiments studying collaborative AR interfaces (J. Chastine & Zhu, 2008). He initially used DART to create a prototype AR system for collaborative molecular modeling. The goal was to allow biologists and chemists to visualize and manipulate large molecular structures in a more natural way than VR. The AR application environment was connected to a molecular mechanics simulation (J. Chastine et al., 2005).




(a) (b) (c)

**Figure 22.** Scenes from the three experimental platforms built in DART (a) the original molecular modeling program. (b) co-located collaboration from the builder's view (c) the guide's view in the follow up study

To better understand how users collaborated within these environments, he ran an exploratory pilot study that specifically studied referencing between pairs. Each participant was assigned either as a guide or builder. The guide could see an existing physical model, and the builder had to construct it under a variety of conditions. Users interacted with the environment using the P5 data glove, and the ultrasonic tracker used for the prototype (the IS900) was replaced with a visual tracker (the ARToolkit) (see Figure 22. (a)). This rapid development and modification was made possible by the use of DART. He created a video sharing plugin for DART so the participants could see each other's POV. He also added support for the P5 data glove. In the pilot they noticed participants used pointing to share information and this informed the two full studies.

The first study was based on use of pointing to convey information. Participants used their fingers as a virtual pointer and the goal was to study how they chose to share information, the accuracy of pointing, and the level of understanding shared between participants (see Figure 22. (b)). From the first user study, TECH found that many guides had difficulty referring to objects, most likely due to the lack of depth cues, such as stereoscopy and shadows (J. W. Chastine, Nagel, Zhu, & Yearsovich, 2007). In 2007 a follow-up study investigated the factors that influence the ability to give and receive references using a virtual arrow (J. Chastine & Zhu, 2008) (see Figure 22. (c)).

Choosing DART

When asked why he chose DART, TECH explained that he had used ARToolkit extensively before, but that he wanted to not have to deal with "the details" if he did not have to. There was a sense of urgency with this project as he was hurrying to finish his PhD, but also that he did not want to "waste time with a more complicated tool." He stated that DART "*lowered the barrier*" for AR and let him focus on the research. He felt that, at the time, there were not many other tools to choose from. Also, he had a relationship with the Georgia Tech team so he knew he could rely on technical support.

Using DART

Unlike some developers, TECH utilized DART for *all four stages in the AR authoring process* (described in Chapter 4) and, therefore, provides an example of a power user who experienced most aspects of DART. During the phase of "Exploring Ideas" with the initial molecular modeling application, the main question he had was "would AR be sufficient way to present what had originally been done via VR?" This prototype was a test of a new tracking system (the IS900) and of the connection between the AR client and the server that performed the molecular bond calculations. He prototyped the molecule app in a "quick and dirty way just to see how AR felt." Later, as he designed the experimental applications he *created simple prototypes throughout the process* to explore possible user interfaces. For example, he prototyped multiple tangible user interfaces for color picking. Initially he explored a virtual color cube that the user interacted with via the P5 glove, but found that it was non-intuitive. He then experimented with a physical marker that a user would move around to pick colors. This led him to the final solution of a tangible slider. He was able to do this prototyping quickly because of the *flexible tracker architecture* of DART.

In terms of "Populating the Virtual World", achieving tight registration was not as challenging as with some AR applications because the virtual molecules were registered to a marker board(s). However, he used the rapid prototyping features and flexible tracker architecture of DART to determine the ideal size of the experimental space for the studies.

There were two elements to his experience "Developing the Application". The basic elements of the applications were *built with the provided script building blocks*, which TECH found to be "laid out in an intelligible way" that "made it easy to see what you could do." The second element was the requirement to create a *custom version of the DART Xtra* (a plugin) for sharing video streams, which was challenging since it was using low level C++ and required knowledge of the Director Xtra architecture.

TECH found "Deployment" (i.e. running his applications on multiple computers over the course of his experiments) to be "*easy and stable*." He commented that it would have been helpful to have the capability to output a secondary view from the applications to the experimenter computer with a customized view for monitoring, rather than just seeing what was on the HMD of the participants. Later, this might have been possible with a combination of the WoZ tools of DART and his video sharing functionality.

Feedback

In terms of the positives and negatives of DART, TECH stated that DART did limit his studies. The main issue was that stereoscopy was impossible, due to *limitations of Director's 3D engine*. This meant that he had to design the visuals with other monocular depth cues to account for this deficiency.

He explained that his biggest challenge in the process was writing the plugin, which was not trivial. Otherwise, the authoring was "just a matter of learning Lingo, and that was just syntax." He said that he was not fond of working with Lingo but that he "got used to it." On the positive side, he *found the timeline helpful* and he appreciated that he

could see the "the visual layout of how the program was going to run" on the Director score. He commented that this visualization of the execution flow was unlike ARToolkit and other subsequent tools.

AR Authoring in the Present Day

In the present day, he has continued to develop AR applications using Android's native SDK and Unity3D with the Vuforia tracker. He finds working with Unity3D adequate and appreciates that it is "in line with how game makers are used to working." He wishes that it provided access to more types of tracking technology and commented that "*there is no concept of physical stuff*" in Unity. Now, he is interested in the ability to "*really bind virtual objects to physical world*." He notes that much of current commercial AR consists of a virtual scene on a marker whereas he wants to create applications that are more integrated with physical elements. He imagines a future authoring utility that would allow the developer to *quickly define relationships between virtual content and physical artifacts* and program the application logic later.

**The AR Novice**

This DART developer was an AR novice (henceforth referred to as NOVICE) who viewed the technology as *a tool to accomplish his creative goals*. He had a computing background, but had not done software development from 1988 to 2004. He was getting reintroduced to technology via a graduate program for which he utilized DART in a yearlong MS project (2005-2006). He was unfamiliar with AR until he saw a video of an AR demonstration two months before he began the project and as a result he became excited about the potential of AR and tangible computing. NOVICE is an example of a DART developer who has some computing background, but is not highly experienced and whose goals are not computer science related. His objective was to create an AR and tangible user interface puzzle-based interactive environment. There is a

wealth of information about the genesis of this project, the struggles throughout, and behind-the-scenes insight into design decisions documented via his thesis and multiple websites as well as a blog that traces the design and development process (Barker, 2006a, 2006b; Barker & Speckels, 2006).

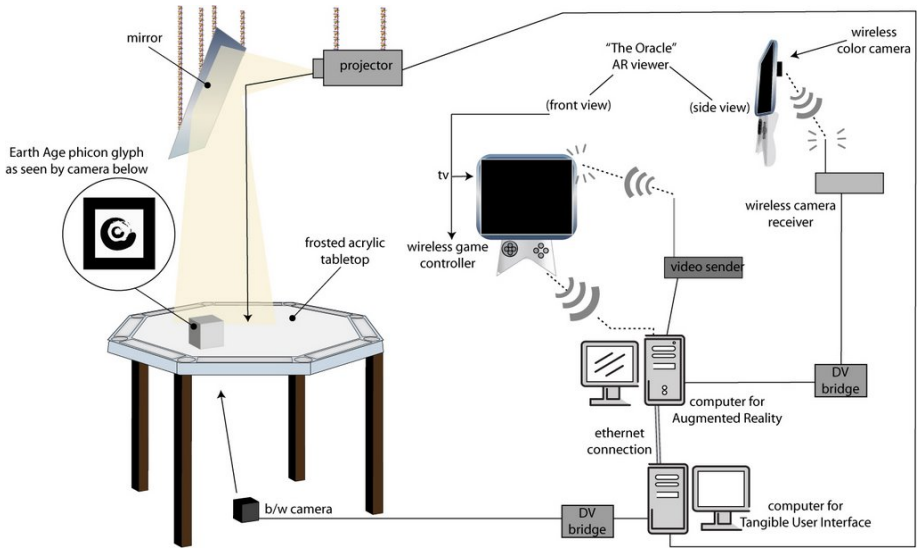
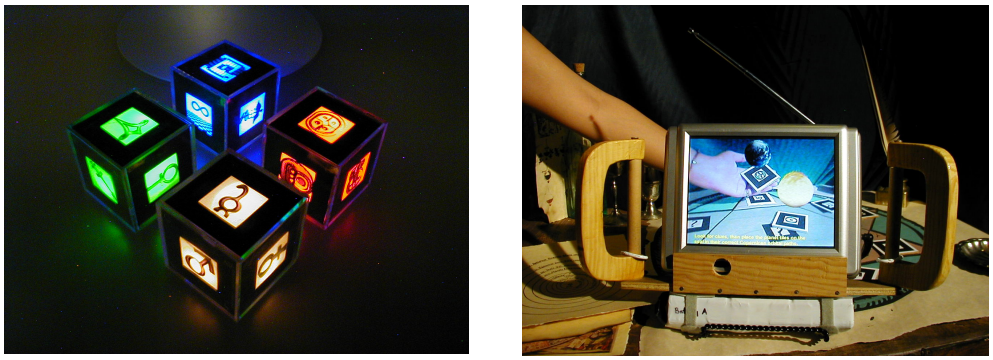



**Figure 23.** The system architecture of "Apollo Beyond"

The Project

The project was called "Apollo and Beyond," an art exploration puzzle played out in real physical space examining man's symbols, cosmologies, and collective unconscious through the ages. The goal of "Apollo Beyond" was to investigate the effectiveness of multimedia design on goal-oriented interactions and collaboration using the combination of AR content and a TUI. The NOVICE and his collaborator built a room that had a central table and four "kiosks" (see Figure 23. ). The center table was a tangible user interface consisting of a projector and a camera under the table tracking markers up through surface. Visitors interacted on the table via physical cubes with LED lights inside and ARToolkit markers on the outside (see Figure 24. (a)). The system utilized two computers, with one powering the center table and one that was running all the kiosks.

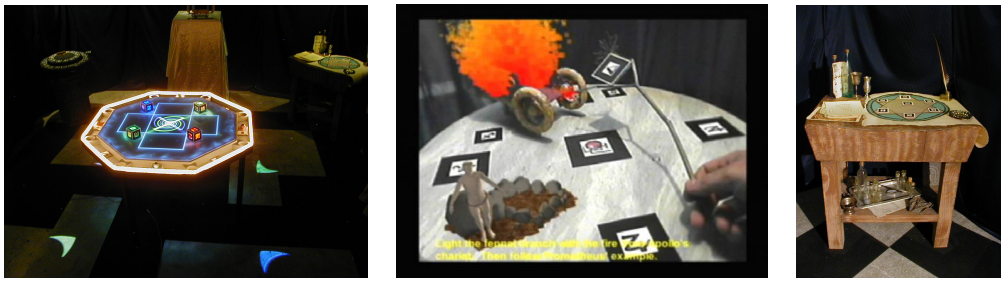
The two applications, called “Omphalos” and “Oracle”, that powered the system were written in DART.



(a) (b)

**Figure 24.** (a) Tracked cubes used on center TUI table. (b) The custom AR viewer

The four kiosks presented the user with AR puzzles to solve. Actions at the kiosks affected the content presented on the center table (See Figure 25. (a)). The users experienced the AR through a custom built personal portable AR viewer (see Figure 24. (b)). Each kiosk represented a particular age in the earth’s history (See Figure 25. (c)). For example, the Greek age kiosk (“Prometheus”) had a physical model of a chariot and told the story of Prometheus stealing fire (See Figure 25. (b)). A visitor would use a branch to interact with a virtual fire particle system; the physical chariot model interacted with the virtual content realistically and, thus, gave the illusion that the physical model was on fire. The goal at this station was to pass the fire from the chariot to a fire pit.



(a) (b) (c)

**Figure 25.** Scenes from Apollo Beyond (a) a view of the installation with the center TUI table and the kiosks (b) An AR view of the Greek age "Prometheus" kiosk (c) The physical installation for the "Copernicus" kiosk (a) Tracked cubes used on center TUI table. (b) The custom AR viewer

The public experienced the installation during one staging. There were 120 visitors who interacted with the experience in groups of six for approximately fifteen minutes per session (Barker, 2006a).

Choosing DART

When asked why they chose DART for the project, NOVICE explained that they investigated other tools including the ARToolkit, but found that they required C/C++ programming. NOVICE did not feel he had the programming expertise required by these tools. When he saw DART he thought "oh yes, I think I understand this. I understand the concepts." He states that DART's model and interface made AR a concept he could grasp and he felt confident he could build his project using it.

Using DART

NOVICE explained that half to three quarters (~ six months) of process was spent learning about the technology, *the constraints, and the design affordances* of AR. Initially, they were unfamiliar with basic AR concepts such as the inherent connection in the application between the virtual and physical cameras and how tracking worked (e.g. "how does the computer even know how to orient the image. We didn't even understand

that in the beginning"). During the design process, their choices were influenced by the capabilities of DART and as they learned it they chose features that they were confident they could implement. They spent significant time playing with ideas, researching DART, and doing *quick technology tests*. Overall they found the physical build was very challenging; from picking cameras, to building the TUI screen and lighted cubes, crafting a wireless video solution, and creating the AR viewers. Approximately another month and a half was spent physically building the kiosks, center table, and decorating the room. Then they built the full system in DART. The collaborator created the art assets and NOVICE did the programming. He was able to *develop the DART applications quickly*. It took ~2 months and he commented, "This is what I loved [about DART]."

They relied on a few key DART features to create their experience. The *capture/playback functionality* was used for prototyping and debugging. They also heavily utilized *VideoActors*, in some cases placing them in multiple layers to create visual effects. NOVICE also leveraged the Director score (i.e. timeline). The two DART applications "would jump around the timeline. Each age had its own. We had a main file that was tracking the phicons. When you put a Greek one on it, it would jump to the Greek file and then it would look for a connection with the kiosk and then jump to the right place in Greek file." What is surprising is that this networked communication between Omphalos and Oracle were achieved by using the *WoZ components* of DART. This was not the intended use of the WoZ feature, but NOVICE explained that it was a method of setting up networked communication that they understood.

The live performance of the system went smoothly and was well received by the visitors. NOVICE stated that during the staging he thought, "This is what it's like to be Steve Jobs. Making things people just get". Following the performance, however, they felt frustrated. NOVICE states, “We spent a year of our lives working on a game that can only be played in one location ever." The project had proved far more challenging than they had initially imagined and afterwards they were “burned out” on AR. However, he

states, "*It did everything we wanted it to. In my mind it was amazing we were able to get all that done. It was a tribute to all the work that was done on DART*."

Feedback

When asked about the positive and negatives of working with DART, NOVICE replied that he found the tool "very well organized" and said he appreciated all the behaviors, cues, and actors that were provided. He also mentioned the *importance of the developer support* we provided (e.g. "If you guys hadn't had that mailing list we wouldn't have gotten off the ground"). Their biggest struggle was with *importing, placing, and debugging their 3D models* (e.g. "we have a 3D model and we can't see anything" and "why aren't we seeing anything!"). Their debugging was aided by a 3rd party plugin for Director that allowed you to inspect the 3D scene graph called 3DPI, but problems with converting and importing models remained. Through trial-and-error they developed a working content pipeline which included exporting from SoftImage to Cinema 4D and then into w3d, the Director model format. NOVICE states, "There were practically no converters. It was like a network of various highways and local streets. Trying to get from one town to another." Once imported some of the models would have errors (e.g. "weird polygons that were out in the middle of space all by themselves…or it would look fine and then you'd rotate and it would have no normals"). Over time they learned to read the raw file format in order to manually fix these problems.

AR Authoring in the Present Day

NOVICE still occasionally works with AR, having created half a dozen commercial projects since then using the FLARToolkit. When asked about his current requirements for authoring he commented that he still needs the same functionality as DART but, in his experience with modern tools, "now it's harder". He finds himself still trying to use DART but Director support is a problem now. In particular, he wishes he

could use DART when he needs to make small AR demos to show an initial idea to a client. Finally, he states about the state of current AR authoring, “things now are so complicated.”

**Artist and Educator**

ARTIST is a professor of digital culture who investigates the future of narrative through explorations of interactive storytelling and interactive cinema. She utilized DART in her AR research and courses over a three-year period. She collaborated on these projects with the technology manager of her lab (referred to as RESEARCHER). Both are *artists, educators, and researchers*. She did not have experience with AR prior to DART, but had done some hypermedia work previously; both she and RESEARCHER were experienced with Director. She was able to found a research lab in 2005 exploring the future of cinema and was granted infrastructure money to buy necessary software and hardware. She, initially, based the lab around the use of DART. This group created, by far, the greatest quantity of projects built with DART between their research and student projects (i.e. DART was used in at least three classes with 15-16 students per over a 3 year period). As fine arts faculty, she explained that she had no workflow for AR research when they began. However, over the years they have become AR experts and have produce many sophisticated AR art pieces as well as their own authoring tools, SnapDragonAR ("SnapDragonAR from Future Stories," 2011) and the ARlab libraries for Max/MSP (Roth, 2011).

Choosing DART

ARTIST explained that they chose DART on which to base their lab research and educational activities because they required a “*point of entry*” for creators who had little or no computer science background. They needed an initial way for the researchers and students to learn to think about AR in an artistic context. They also were interested in

using a non-vision-based installed tracking system (the IS900) and therefore could not use the other approachable authoring tools of the day such as ARToolkit. The funding for the lab supported the purchase of the tracking system and a sophisticated HMD. Therefore, their early AR work was more focused on the use of head and hand tracking rather than marker based tracking. While they have gone on to author with a variety of tools such as FLARToolkit, Max/MSP, Metaio's Junaio, and their own custom tool, SnapDragonAR ("SnapDragonAR from Future Stories," 2011), ARTIST commented that "*DART saved our lives*."

Using DART for Education

It was a challenge to develop a lab workflow based around DART. There was a learning curve for ARTIST, RESEARCHER, and the students as they got started with DART (e.g. ARTIST commented, "I wouldn't say it was easy"). She explained that by the time her lab was established the initial impetus for basing DART on Director, because it was widely used by technical designers and artists, was no longer true. At that point, DART was not leveraging workflows that "artists do anyway." Some of the students were enthusiastic if they knew Flash and the majority of film production students adapted, but some were resistant. There were enough obstacles to student development that the dream of rapid iteration was not fully realized.

Both ARTIST and RESEARCHER focused their efforts on helping the students understand AR technology and the potential for using it artistically. They found that, for the students, it was critical to see working AR systems and to, early on, *create a simple application by themselves* (e.g. ARTIST explained, "Seeing something working with a tracker was magical, not arduous", "Getting a foot into augmented reality let them know they could be contributors", and " it let them imagine different ideas and limits of technologies"). The students were often surprised to find that even with all the equipment and technology involved, there were still *considerable constraints* on what could be built.

ARTIST and RESEARCHER found that it was a three-week process for a student to learn enough about AR and DART to achieve an early prototype. A challenge was that few had scripting experience and they did not know Lingo. At times, the students would initially envision experiences that were very interactive (e.g. RESEARCHER described, "move your hands in a circle" to trigger a behavior), but when they realized it would take scripting then they would abandon the idea. However, this process helped them to think about natural interfaces. ARTIST wanted them to explore interaction and behavior. The students did not have a lot of technical skills, but they did have "stories" to share and skills that could be translated, such as communications and film.

To help the students get started they would provide a *very basic DART example template* that included a few hotspots such that when a tracked hand entered the zone a DART event was fired. They also utilized a relatively simple DART application they had built, called "Fortune Teller." It displayed a video sphere; the user would pick up a tarot card and put it in a bowl, which would trigger the application to choose and play a fortune teller video clip. They also made a series of *video tutorials* to get students acquainted with basics such as creating an Actor and importing a texture. They found these examples were instrumental in getting students comfortable with working in DART. They also referred to the code for FAM to understand how a big DART application architected. They would have benefited from *other examples of large-scale projects* to help the students understand how to design such a system. Overall, the goal was to have the students thinking long term about AR and where the medium where will go next. For their term-long projects the students were encouraged to "think big, but make small."

Overall, there were differences in AR authoring requirements for education rather than a pure production setting. In production, the objectives may be efficiency, speed, and quality of work. Large ambitious projects most certainly involve a separation of duties where project participants may not directly interface with AR authoring. However,

in this situation ARTIST and RESEARCHER felt it was crucial that their students be forced to confront all aspects of AR development. They pointed out that the *approachability of DART*, which allowed non-technologists to *directly engage with AR technology* supported their educational and research mission, which they felt dovetailed with the maker movement and collaborations in digital media. In these communities, small teams work iteratively to push the creative envelope. ARTIST pointed out that much of the AR community now is saying, "What is the killer app?" seeking commercially viable deployable applications, while in her community the impetus is "I want to explore this [new medium]."

The Projects

The students in the courses created a large number of projects throughout the years. Typically the students were in theory heavy programs where they were not building artifacts. The ARTIST stated that DART let her do something different with her courses, which was key as "physical space is so different from the digital palimpsest" and "*DART was an object to think with*."

Early on, the students would create MR experiences that did not require tight registration with such as an "MR Mirrors" project that placed videos faces around the user's head (see Figure 26. (a)). They also built DART MR applications inspired by the novel "Invisible Cities" by Italo Calvino. These types of projects helped them become familiar with the *technologies and experiences afforded by AR*, but it freed them from having to design for a particular space or concern themselves with physical props. ARTIST stated that this resulted in "a lot of play, because it was hard to have failure in that environment" and "*DART was forgiving, letting people start with baby steps*."

(a) (b) (c)

**Figure 26.** (a) the user's view of the MR "Mirrors" student project (b) A "Multi-Sequential Poetry" user reaches for a augmented cube (c) a frame of video from the "Augmented Cambodian" project

The potential of DART was more fully explored via the ambitious sophisticated projects built by students in collaboration with ARTIST and RESEARCHER. For example, the project "Multi-Sequential Poetry" explored what happened when a poem is made navigable and interactive (Skolnik & Roth, 2008). This experience placed audio segments of Christian Bok's poem "Vowels" on thirteen boxes hanging in the installation space. As users reached for a box they heard the associated clip (see Figure 26 (b)). This allowed users to interactively explore the more than six billion permutations of the poem. The "Augmented Cambodian" uses augmented reality to reveal untold stories of survival during Khmer Rouge (See Figure 26 (c)) (Walker, 2009).

"The Coatcheck" explored migration, memory, and post-colonial theory (see Figure 27. (a) and (b)). The installation was composed of elements such as a coat rack with a single coat hanging on it, a fog screen with scenes from a train station, luggage filled with travel documents, and a window with a projection of a coat turning in circles on a coat rack. It was exhibited to the public as recently as May 26, 2011 (Koc & Cheng, 2011).

(a) (b) (c)

**Figure 27.** (a) The "Coat Check" project open in the DART environment (b) A user interacts with "Coat Check" (c) The augmented view of "52 Card Psycho" originally prototyped in DART

Using DART

Reflecting upon the lab's use of DART, of particular interest is how they *developed processes for AR design* and the obstacles they faced due to the backgrounds and existing workflows of their students. They found that those who worked in time-based media (e.g., film) *struggled more with understanding the possibilities of AR*. The theoretical students had less trouble due to their lack of preconceptions about technology and their technical ability. ARTIST explained that film students often initially made projects that were focused on presenting 2D content in squares (i.e. "Using [AR] like a television"). ARTIST described, "Some students struggled imagining new things. *They couldn't think in terms of the physical world*." She found that it was crucial for the students to develop their own, very simplistic AR application, as soon as possible in order to foster the development of a correct mental model of how AR worked.

The theme of the class was "coming to understanding through making" and this, coupled with their experiences using DART, led to the creation of their custom SnapDragonAR tool. The goal was to create an authoring tool that was even simpler than DART. *Inspired by the use of rapid prototyping and video in DART*, but wanting students to be able to create their first application in minutes, they built an environment that distilled the authoring process down to a single function that could be controlled

solely through a GUI interface. This functionality was the ability to place a video texture on a single marker. They found that the act of creating this very basic application helped the students understand AR. Due to their backgrounds in film this example exposed the students to a feature of AR that is diametrically opposed to film, the fact that the creator loses control over the viewer's POV and, thus, cannot completely control the experience through editing as they normally would. ARTIST stated that this fact "blew their minds." She believed it was very important for their students to "speak the same language of AR" and to understand how to requisition their projects. Much of the work that went on in the classes was to help the student *explore what aspects of cinema could be leveraged in an AR experience* and which could not. In some cases, they found that the "perfectionist" tendencies of film students, accustomed to absolute control over the user experience, would negatively affect their ability to complete their AR designs.

When asked about their use of the more advanced DART features, intended to ameliorate the AR specific authoring challenges related to physical space, rapid prototyping, and debugging, ARTIST explained they "*got in a habit of not using them*" because their applications "*were not sophisticated enough to need them*." Over time the lab research process evolved such that no one bothered with features such as capture/playback because the feeling was that it was quicker to "just build in DART and then try it." They also did not face many of the typical challenges of authoring for remote and/or very specific physical locations or objects as they were typically building applications for their empty lab space, which contained the IS900 tracker. In a way, their authoring needs were more closely related to VR than AR. RESEARCHER explained that he believes the advanced prototyping and debugging features were valuable, but just not for their purposes. He felt they would be essential at a point in the development pipeline that their applications did not usually reach. Surprisingly, when questioned about the WoZ features, ARTIST commented that she had always assumed that DART documentation that referred to them was simply describing the HCI technique that could

potentially be leveraged via DART, and she did not realize that there were actual DART components premade to support WoZ applications.

Feedback

In terms of positive feedback regarding DART, overall both ARTIST and RESEARCHER expressed gratitude for DART. More specifically, they stated that *their own tool-building work has been informed by DART*. For example, they described how their Max MSP libraries for AR authoring take the concept of common behaviors (i.e. AR building blocks) from DART and other toolkits and simplifies them such that they can be used without any scripting (e.g. Arlab.tracker.recv receives transformations from trackers and Arlab.hmd.video puts live video in the scene) (Roth, 2011). This work was an evolution toward a tool even more approachable to non-technologists than DART, leveraging the Max/MSP model of graphical manipulation of patches, which can be linked together and controlled with toggle switches and menus.

RESEARCHER commented that he often wanted to use a standard authoring approach, which is *pasting together components from previous projects* to get started with a new application. Unfortunately, this was a difficult process due to Director's restriction on opening more than a single project at a time on a computer. Because of this, it was difficult to refer to an example or an older project while working on a new one. He also found that *the handling of video in DART* frustrated the film students. They often wanted to build their experience around very large high-resolution video files and the process for creating VideoActors was tedious due to limitations in Director's 3D engine and video playback support. He commented that they still encounter these same types of problems in Unity3D and have had to develop video texture solutions similar to DART. They also struggled with 3D model pipeline. Both commented that if *conversion of their content* had been easier, if they had been able to utilize original versions of assets more often, then that would have been a huge boon for their workflow.

RESEARCHER felt that students were confused by DART's use of the timeline. He found that he and the students understood the original time-based Director model, but struggled, conceptually, with DART's use of application segments on the timeline that were logically separate, but were not controlled by the Director clock. Also, the concept of "actors" in DART made sense, but the fact that these actors were containers that could contain modifying scripts (e.g. triggers, transforms, movement etc.) would sometimes confuse students.

Lastly, many of their challenges were related to debugging tracking related problems. Specifically, they struggled to determine whether an application was not working as expected due to hardware problems with the tracker, a failure of the transport mechanism for tracker updates, a misconfiguration of the VRPN server (which could be internal to DART or run externally) or client, a bug in DART, or a developer error in the DART project. RESEARCHER summed it up as "*not enough high level access to low level debug information*." He admits that at times frustration over unexpected behavior would be blamed on DART, when it fact it was due to hardware failure or developer error. For example, they struggled with confusing tracker performance for quite awhile until a knowledgeable visitor to the lab commented that one crossbar of their tracker was clearly broken. They also blamed DART for a lack of registration precision, until they realized they had mounted the tracker sensor in the wrong place on the HMD. These struggles were due to the fact that they were new to AR. There is a considerable learning curve required to become proficient, not just with AR software development, but also with the sophisticated esoteric hardware that is often part of such systems. DART did not provide enough transparency about all elements of the system that would have helped them *identify the source of errors*.

In conclusion, ARTIST and RESEARCHER commented that DART "just took us awhile to learn. But then it just worked" and then "*the workflow was obvious*." They

explained that, looking back, now that they have built similar tools they consider DART "wonderfully elegant."

AR Authoring in the Present Day

ARTIST observed that her students have changed. Now, the incoming students are familiar with the concept of AR. However, some of their authoring challenges have remained the same. For, example the film students are still in need of a *fast pipeline for creating and placing video* in an AR prototype. Although Unity3D (and the UART extensions) handle content loading more effectively they are still struggling to create that very efficient pipeline which does not require them to make artistic concessions.

They continue to be focused on working with high-end systems that allow them to create applications with tightly registered augmentations, presented through HMDs, controlled via natural and tangible interactions. Overall they are not interested in outdoor mobile AR applications that use mobile devices for display and GPS for tracking, which are common in the commercial world. Therefore, they have spent many years contemplating an ideal AR authoring tool for their specific needs. This has included experiments with their own tool building and modern tools such as String ("String," 2012), Vuforia, D'Fusion, and Unity3D.

Over time they created a "wish list" of capabilities. However, they now realize that this list was heavily focused on *low level technology needs* (e.g. better content loading and sophisticated image-based tracking), many of which have very recently been met by the aforementioned tools such as Unity3D. However, they still have goals regarding how best to *teach AR authoring* and to *explore the medium artistically* that resulted from their work with DART. They want to rigorously study spatial story telling, but they, and others, have yet to achieve the critical mass of creators, projects, and experience. A frustration for them is that their work is often bound to their lab due to the technology requirements and as a result their art cannot be shown at a gallery and it

cannot be experienced by a large number of people. They feel that these larger issues regarding how to create authoring systems that do not limit the creativity of the artist, that allow them to work with a variety of high quality content efficiently, that support the design of novel and thought-provoking experiences which can be widely exhibited are still yet to be addressed.

**The Theater Director**

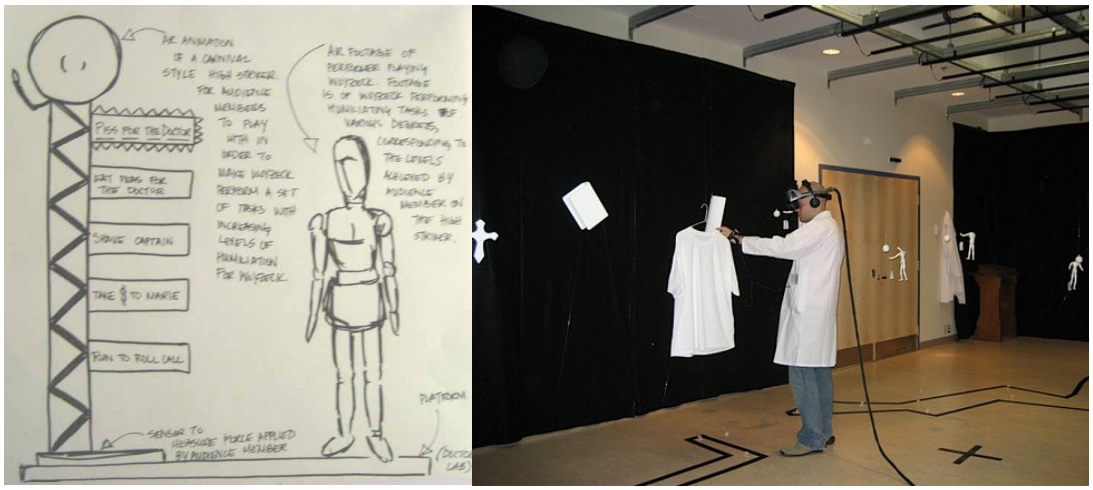
DIRECTOR is on the opposite end of the spectrum from TECH. Her background in theater is as a *director and playwright*. Her interest in AR is to use it to explore digital technologies in story-based theater. As a graduate student of ARTIST she created one of the most ambitious experiences built with DART. However, she was entirely focused on design and art. She did not engage directly with DART, but was the driving force behind the entire production including the AR design. From her point of view, there is too much focus in new media on technical skill rather than on the art.

Her motivation was to make an AR version of a play that contained affordances she associated with AR such as "responsiveness" and "media in space." Her collaborator, DEVELOPER, who had a web development background (including Director experience), handled the DART programming. DEVELOPER was interviewed as well and her comments are included in this section. The team included a composer, a choreographer, a film lighting expert, singers, dancers, a technical director (programmer), and a documentarian.

The Project

Over the course of two years, DIRECTOR and her team created and staged an AR adaptation of the play “Woyzek” by Buchner (Rouse, 2007; Rouse, Lee, Padgett, & Shepard, 2007). This play was well suited for AR because the order of the individual scenes is unknown due to the playwright’s untimely death. Arranging the scenes in

different permutations yields a different story (e.g., a suicide, a love story, a murder etc.). DIRECTOR thought that an AR treatment would allow the audience to encounter scenes in random order and construct their own version of the story. Also, she felt that a representation of the characters as ghostly would suite the technology because "dead people worked well in AR"; they could be visually fuzzy, transparent, and not tightly registered with the world. The work to stage this play included creating an original English translation of the play from the German, composing the music and lyrics, choreographing the dances, gaining the knowledge to design and build a chroma-key studio, directing the cast of singers and dancers, editing the resulting video, designing and constructing the physical set, as well as the design of the user experience and software development for a medium which no member of the team had previous experience.

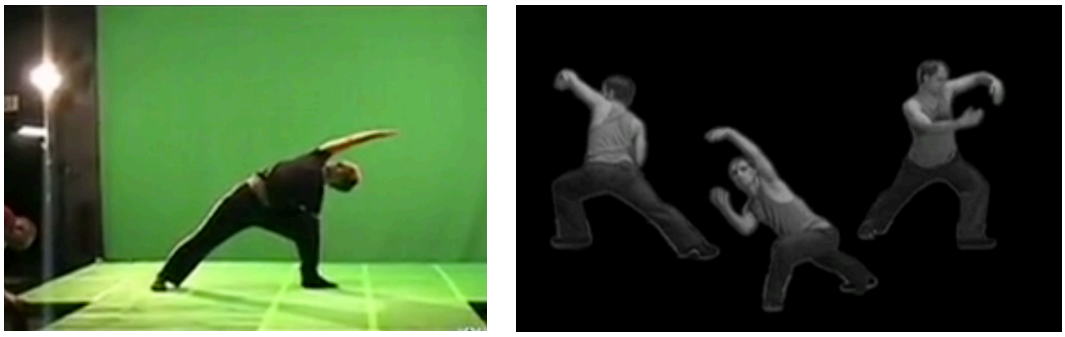



(a) (b)

**Figure 28.** (a) A sketch showing an early vision for the doctor interface (b) a visitor experiences Woyzek AR experience.

The technical setup for the application included the IS900 tracking system, which supported the precision interactions via a head and hand tracker and an optical see-through NVis HMD. Users would wear a lab coat as they went through the experience; it

served a thematic purpose and also could hold the battery packs required for the HMD (see Figure 28. (b)). The play was staged in the AR lab space and consisted of blacked out walls, props hanging in space, and regions delineated on the floor via tape. The AR content consisted of *VideoActors* and *AudioActors*. The space was populated with physical objects, which created a *tangible user interface*. The objects were manipulated by the user (e.g. placing heads on wooden figures, stabbing a mannequin in various locations) to trigger AR scenes that were registered with them (see Figure 29. ). The mannequin represented Woyzek; bulls eye targets on the mannequin invited the visitor to play the role of the malicious doctor who tortured Woyzek (see Figure 28. (a)). The visitor would stab a target with a stage knife to perform an experiment (i.e., trigger a particular scene).



(a) (b)

**Figure 29.** (a) A dancer performs as Woyzek on the green screen stage (b) a frame of video from the Woyzek AR experience.

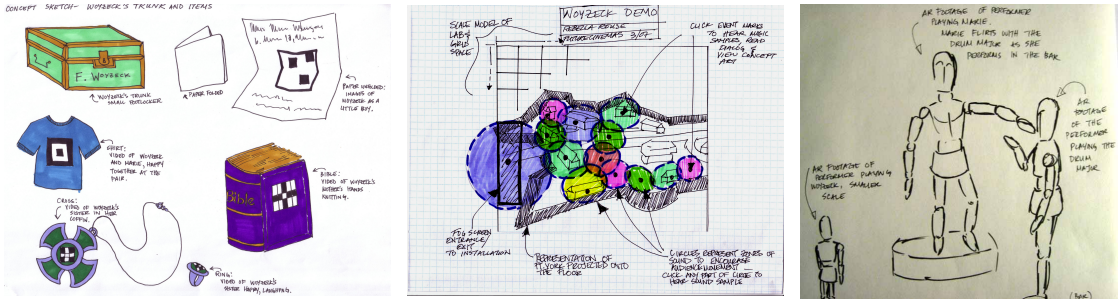
Woyzek was exhibited for one week and approximately 50 people experienced it. DIRECTOR shared anecdotes from the exhibition. In general they received positive feedback from visitors. DIRECTOR commented that visitors exhibited behaviors that suggested they felt engaged and immersed in the content. For example, at one point a visitor placed a head on the doll and screamed when the augmentation appeared. In

general, however, DIRECTOR was frustrated by the individual nature of the experience (i.e. one visitor at a time, who was limited in movement and FOV by a tethered HMD). While her original goal was to create an AR version of a play, in retrospect she does not feel that Woyzek had the important participatory social aspects of a play that comes from live performers acting in front of an audience.

I interviewed PROFESSOR who was one of the visitors. He shared his impressions of the experience. As a researcher who works in AR he understood the complexity of the experience the team had created. He commented, "It was amazing that everything worked." He also explained that he was most impressed by the non-AR aspects of the experience and was struck by the high production quality. He found the material appropriate for AR presentation due to the surreal nature of the play, which lends itself to avant-garde production. He explained that while he was experiencing the dance and music content he felt as though he was watching an “off Broadway” production that had been transformed into AR. During the experience he was thinking, "I wanted to see the play. It was tantalizing. They were literally dangling the content it in front of you with props." He found the passive components of the experience more fun than the interactive. He felt that the AR was almost a separable layer from the content which was "eerie," "uncanny," and appropriate for the play. He echoed DIRECTOR’s sentiment about the “usual problem of the individual of nature of AR due to the state of technology” which he felt affected the nature of visitor experience.

Using DART

DIRECTOR and DEVELOPER described their design and development process, highlighting the role DART did, or did not, play in each.




(a) (b) (c)

**Figure 30.** (a) Sketches showing initial ideas of how marker tracking could be used in the experience (b) a map of the physical space for the experience (c) A sketch representing an AR scene in Woyzek

The initial brainstorming stage of the project was executed with little or no technology. The lab was still under construction and the researchers had not even begun to work with DART at the time. Therefore, DIRECTOR relied on *traditional ideation techniques* from her theater training. She created a vast number of *sketches* to explore possibilities for the physical and the augmented experiences. Later in the process she utilized layers of transparencies to approximate the AR experience. Figure 30. shows example sketches ranging from early investigations of tangible interface options using marker tracking (a), to drawings created by the DIRECTOR and choreographer to plan the video augmentations (c), to schematics from later in the process that define the physical layout of the exhibition space (b).

Early on in the "populating the virtual world" stage the team still did not have access to DART. They shot the video for the augmentations (see Figure Y) without ever having used DART or exploring how the VideoActors worked. However, DIRECTOR had experienced FAM so she had a general idea of what was possible. At one point they mocked-up the experience using an image map in HTML; the user could click on the map to trigger the AR scenes. DEVELOPER described this process as *participatory*. She approached the design process from the *technology direction* while the DIRECTOR focused on *artistic goals*. DEVELOPER spent time exploring the capabilities of the

hardware and software (DART). She would share her knowledge regarding what was technologically possible, which would inform DIRECTOR's artistic choices. DEVELOPER explained that she would often "shoot down" DIRECTOR's ideas during brainstorming, and then they would keep talking until they formulated a plan that DEVELOPER believed was possible. Once they were able to experiment directly with DART, the DEVELOPER would create example DART projects in order to for DIRECTOR to experience features directly. They often would *walk through the physical space to share ideas*. In some cases they would use the tracker during these sessions in order to *capture key locations* for later use in the application. Each AR scene was placed in a physical zone in the space, to be triggered when the hand tracker entered the zone. DIRECTOR would walk around the space triggering the scenes, communicating with the developer who was in the other room; this was how they calibrated the zones.

The application development of Woyzek differs from the previous DART projects discussed in this chapter, as in this case, the project leader did not engage directly with DART. DIRECTOR explained that she would routinely "check in" with DEVELOPER to test the experience. They had sessions for *user experience testing* and *debugging*. DIRECTOR commented that she would give feedback during these sessions, but that it was hard for her to understand the *technology constraints*. DEVELOPER commented that over time she simplified the initial ideas so that the application was based on position triggers, which generated cues that controlled VideoActors. She utilized the score to spatially group components of the application. She explained that she relied on *modifying existing scripts* as the foundation of her application. She said that a few times she "*tried starting from scratch and wasn't very successful with it*." By the end of the project, however, she realized that these versions of the scripts had been significantly modified from the originals.

When it came to deployment of the Woyzek exhibit both DIRECTOR and DEVELOPER felt that it ran *smoothly* and *reliably*. DIRECTOR was pleased that the

visitors were able to interact with the application with little guidance. Both also mentioned how interesting it was to watch the real-time playback of what visitors were seeing as went through the experience. DIRECTOR was intrigued because it was as though the visitors were "editing the film with their bodies." They did not utilize capture/playback, but they discussed using it to capture a personalized linear version of the narrative to share with visitors after their experience.

Feedback

In general DIRECTOR found the project very challenging. When DIRECTOR finished Woyzek she said that she vowed, "I will never work with AR again." She found the project to require far more effort than initially anticipated. She explained, "*every step, there was so much more to learn and so much to do*. From how to deal with the video to building the stage, taking it to the site, painting it [for Chroma keying], renting the light kit, figuring out how to light it [the stage], how to shoot it [the video], borrowing a camera, learning how to export it [the video], editing it [the video]. It was more work than I ever would have imagined." There was conflict between what she wanted to do versus what was possible. There were many *unknown constraints* until she learned more about the technology and the medium. She explained, "It was like a lobster in a pot. You didn't know how hard it would be when you started and the complexity built slowly and then it was too late to turn back." While her experience with DART certainly contained frustrating moments, however, she explained that DART was a positive aspect of the process. She commented, "*there was no other way we could have done this project at that time*" and "it was amazing that it [Woyzek] worked"

The main criticism from DIRECTOR and DEVELOPER regarding DART was the challenge associated with *importing video* as individual frames. However, the DIRECTOR also shared her high-level vision of an ideal DART interface. She would have liked an authoring environment that allowed her to *engage more directly with the*

*technology*, "I would have been involved in the programming process, despite no exposure or aptitude for programming." She imagined a drag-and-drop interface that would have supported early prototyping of the project through the "first step," which she defines as the *placement of proxy content at physical locations*; basically, tagging locations to receive future content. She described a simple authoring tool she experienced prior to DART that was part of the "Mobile Bristol" project ("Mobile Bristol | DShed," 2005), which was used to create location-based audio experiences. The interface was very simple. The user was presented with a 2D map of the physical space on which she could place audio clips. DIRECTOR found it easy to use and intuitive. She realizes now that it was not flexible and was only for sound, but at the time she assumed that her AR authoring experience with DART would be similar. She explained that she would have liked to have the opportunity to do early prototyping of Woyzek via this type of 2D map interface.

AR Authoring in the Present Day

When asked about current AR applications based on mobile platforms, using GPS or 2D markers for tracking, DIRECTOR stated, "current AR is boring for art." However, she is currently working on AR applications using the Argon browser (see Chapter 7). In these projects her focus is on creating content (e.g. script writing) related to historical tours. She explained that the expectations and goals differ from Woyzek. She finds the scale of the projects more manageable. However, she found the types of AR experiences made possible by high-resolution tracking and HMDs to be very compelling. Ultimately, she believes that control via *expressive user movements* are critical for effective AR applications and that this input should affect aspects of the physical environment (e.g. "moving robots or changing the lighting in the room"). In a similar vein, DEVELOPER commented that she has worked on outdoor GPS-based AR applications since Woyzek as well. And, although she finds the hardware much more accessible now, she explained that

after having worked with the IS900 she did not find these types of applications to be very exciting.

When asked about her ideal AR authoring tool, DIRECTOR described a system that assumes *collaboration amongst stakeholders with disparate skills and foci*, which is critical in theater. In her roles as script writer, director, and producer, for example, she wants to be able to prototype with little technology, while others on the team work directly with the technology. She commented that, "*artists may not want to program*." She commented that in digital media education now you are told "everyone needs to understand the technology well enough to collaborate," but she feels there is too much focus on working alone. She discussed how "great performances like the Olympic opening ceremonies have lots of participants in the design and production" and "I am never going to be the lead programmer on a project. Not if I want it to be good"

## Reflections on DART and AR Authoring

In this section I discuss the interviewees' reflections regarding DART as well as the current and future state of AR authoring needs (see Appendix I for coded data). These insights are drawn from the previous four highlighted developers/projects as well as additional feedback gleaned from interviews with two other DART developers (RENAISSANCE, a very technically proficient digital media artist, and MUSICIAN, an experienced AR designer with limited programming skills who has built AR art pieces including some based around interactive music). The insights regarding general AR authoring and workflow are also drawn from interviews with three AR experts. PROFESSOR has worked in AR research from the media theory and humanities side for over a decade. He provided insight into how digital media students learn about AR and his perspectives on the authoring needs of technical designers. STUDENT is a graduate student in Human Centered Computing, researching AR authoring and multi-scale design. He also collaborated on an AR art installation as technical lead; he provided anecdotes regarding the challenges of collaboration between technical and non-technical designers. Lastly, MANAGER was the producer and project lead of the AR enhanced Duran Duran Project described in Chapter 3. He discussed the unique aspects of collaborative AR design with performers well outside the traditional digital media realm.

### DART Feedback

The goal of this section is to capture the significant feedback regarding DART that was gleaned from the interviews as well as anecdotal feedback from other DART developers gathered, over the past nine years, via face-to-face conversations and threads on the DART mailing list. I will highlight common DART comments and critiques in the categories of the *Director platform*, *accessing technology*, *debugging*, *rapid prototyping support*, and *community*.

Overall, the full range of developers, from highly technical to design oriented, appreciated the fact that DART simply made their AR project possible. It was striking how often the interviewees uttered the exact phrase, "it was amazing it [the project] worked". The developers appreciated the *lower barrier to entry* provided by DART. Multiple developers mentioned a sense of urgency in completing their project and that DART met this need for *rapid development*. There were also many comments regarding DART's *stability* and *reliability* during deployment. It was used successfully to run a range of "real" systems for trade shows, user studies, and art installations. Over the years I have witnessed many interactions where people have thanked us for creating DART and there are dozens of examples of successful projects built with DART. I believe that it is a testament to DART's utility that it is still downloaded years after the last release. Many of my interviewees expressed desire for DART features in modern tools. RESEARCHER and MUSICIAN acknowledged DART for *informing authoring tools* they have developed since.

Director as a platform

Unfortunately, Director was already on the wane when DART was released. As a result, DART was never able to leverage a vast and highly energized developer community as we had hoped. Many of the interviewees did not mind using Director, but it would not have been their first choice, and some were resistant to learning it. Few identified themselves as Director experts, although they represented a range of previous experience from complete novice (TECH), to passing knowledge (RENAISSANCE, DEVELOPER), to having used Director in the classroom (ARTIST, RESEARCHER). A weakness of DART was the requirement that scripting in Director be done in their proprietary Lingo language, which no one professed an affinity for.

Some of our DART design choices were precipitated by Director conventions or technology constraints. For example, the interviews revealed that DART's *hybrid use of*

*the timeline* was powerful for some (e.g., TECH, NOVICE, DEVELOPER) but was frustrating and abstruse to others (e.g., RESEARCHER, RENAISSANCE). I believe there was value in having a visual layout of the application, however, we were constrained to a time-based metaphor that was not entirely appropriate for the DART architecture.

The value of *code reuse* has long been acknowledged (Detienne, 2001) and environments such as Smalltalk have been designed around supporting reuse (Rosson & Carroll, 1996). Therefore, it is not surprising that multiple interviewees commented that they preferred to start a new application by *cutting and pasting* elements from older projects. While it may seem like a trivial constraint, the inability to easily select and copy elements from the timeline of one Director project into another significantly impacted many of our developers' workflows. ARTIST and RESEARCHER reported that this affected their ability to teach DART and reduced their ability to have students leverage existing sample applications. This is still a shortcoming of certain modern tools such as Unity3D as well. Reuse is easy when the applications consist entirely of text source code. However, when components of the application are defined in a visual editor, when the author creates scene graphs, or lays objects out on timelines it is not as straightforward to "copy and paste" parts of that project. There is a need to recognize this requirement and craft solutions for future high-level tools to support reuse.

Three technology limitations of DART and Director were responsible for the majority of developer struggles. First, the most common technical support issue the team dealt with (based on DART mailing list posts) was with developers *configuring cameras*, for video-mixed AR, to work inside of DART. This was due to limitations of Director and of our DirectShow based plugin. To use a new camera, developers had to identify configurations that were supported by DirectShow via a list of modes printed to the console and by then transferring the appropriate configuration text string into the LiveVideo property page of their DART project.

One of biggest complaints observed in the interviews (and also evident in mailing list) was the frustration associated with *exporting and loading of 3D content*, which was due to the lack of models and exporters for the Director w3d format. Some commented that, while this is better in modern tools like Unity3D, it is still a problem with modern game engines.

While more technically adept developers (e.g. TECH and RENAISSANCE) relied on 3D models for their applications, it was clear from the interviews that the design focused developers (e.g., ARTIST and DIRECTOR) heavily utilized the VideoActor functionality to construct their experiences. It appears that our focus in DART on support for video based augmentations was a strong point of the tool for technical designers and artists (e.g. ARTIST and RESEARCHER concentrated their students almost entirely on the use of video-based augmentations). However, there were universal complaints about the need to *export video as individual images* for import into Director. Unfortunately, this was the only technical solution for Director at the time. ARTIST and RESEARCHER mentioned they have had to use similar approaches to support video augmentations in Unity3D due to the lack of video texture support for mobile.

The most common critiques of DART from users were due to interface and design decisions forced upon us by Director (e.g. the tedious method for loading videos, proprietary 3D format, and scripting via Lingo). None of the interviewees felt that having the tool inside of Director provided value. Some were neutral (e.g., NOVICE, DEVELOPER) but some were decidedly negative towards Director (e.g., RENAISSANCE). Therefore, as we reflect on DART we must consider whether it was wise to base our authoring tool on an *existing piece of commercial software*.

If we had not been dependent on Director, DART might still be a viable tool. It was the lack of updates to the 3D rendering capabilities of Director that caused DART to become outdated and for us to discontinue development. There appears to still be interest in authoring with DART. I found multiple examples of developers who are still trying to

use DART even though there has not been a new version of Director in years. If we had complete control over the application we, theoretically, could have continued to release new versions.

However, I believe that, ultimately, DART was only possible because we leveraged Director. Indeed, there were problems with integrating AR into Director and certain features were awkward as a result (e.g., the use of the timeline), but I think that our users may not realize what value it provided. A small team of three developers in an academic environment could not have produced a custom tool from the ground up with the range of robust features that Director provided. There certainly were custom-built AR authoring tools available at the time of DART (e.g., ARToolkit, Studierstube, and ImageTclAR), and yet our users did not choose them. In the previous section there were multiple comments on how *DART was the only tool they could have used*; they complimented the low barrier to entry. They were surprised they were capable of creating an AR application with a high the level of sophistication. It was our use of Director that allowed us to provide these features.

Our use of Director may have sacrificed longevity for the project, but because of it we were able to create a useful tool for a moment in time (i.e., 4-5 years). As a researcher it is important to contemplate whether it is possible to leverage an existing authoring environment or platform without facing the problems of obsolescence. As we learned with Director, it is difficult to predict what the dominant authoring tools will be over a span of many years. Many of my interviewees commented that, at the time, they wished DART had been built in Flash, which was not possible at the time due to lack of 3D support, but now Flash is already waning. Meanwhile *web technologies* such as webGL and HTML5 are waxing and a new breed of AR browsers, such as Argon, based on these standards are appearing. Another option for modern authoring is to leverage *game engines* such as Unity3D. Companies such as Qualcomm, Total Immersion, and ARToolworks have produced plugins to expose their tracking libraries into Unity3D.

Ultimately, however, I think we must acknowledge that computing is a rapidly evolving field and that a widely used programming language or IDE of the moment may be obsolete in a few years. The important thing is to have a general understanding of AR authoring needs, informing conventions for authoring that can be continually recreated in new tools.

Accessing Technology

Toolkits from other domains have previously recognized the need to provide *high-level support for hardware access* (Dey, et al., 2001; Klemmer, et al., 2004). Greenberg et al. describe their developers prior to phidgets as "immersed in a quagmire of tediousness" (Greenberg & Fitchett, 2001). Similarly, in the Papier Mache project they recognized the need for technology portability (Klemmer, et al., 2004). Their interviews with users revealed that acquiring and abstracting input was the most time consuming and challenging part of application development. The interviewees requested a more flexible method of defining associations between input technologies; so that they could easily explore different options. For example, in their system a project can be prototyped with computer vision but then deployed using RFID. In the AR and VR domain, OpenTracker provided a "write once, input anywhere" approach to managing hardware based on XML (Reitmayr & Schmalstieg, 2001). Inspired by these tools, DART also allowed developers to access a variety of trackers and input devices (e.g. game controllers, Intersense trackers, phidgets, motion capture systems etc.) and to flexibly swap between solutions via high level authoring.

TECH, NOVICE, ARTIST, RESEARCHER, DIRECTOR, and DEVELOPER all cited this capability as a significant asset in DART. This allowed TECH to *flexibly switch* between the IS900 head tracker to marker tracking in his prototype and subsequent applications. He also commented that he appreciated that he was able to *add support* for a new device, the P5 Glove, to DART. NOVICE was surprised at the ease with which they

*integrated game controllers* into their AR interface. ARTIST and RESEARCHER explained that high-resolution head/hand tracking was critical to their AR work and that DART was the only option that made this technology *accessible to the students*. Multiple interviewees commented that current tools are focused on vision-based tracking and as a result they find themselves searching for a modern DART alternative.

Debugging

The interviews showed that the more technically experienced developers such as TECH and NOVICE developed *effective debugging strategies* within DART. TECH explained that during development he would often isolate units of functionality and focus on them one at a time. For example, he described working on the glove support while actually wearing the glove. He could quickly make small changes to his DART project and immediately experience the results. NOVICE described how the use of the 3rd party scene graph inspector 3DPI helped them solve many of their DART problems, which were related to 3D objects not having the expected appearance. He also learned to read the raw 3D model files in a text editor and edit them manually to remove erroneous polygons. While novice was inexperienced with AR he had a technology savvy that helped him investigate and solve his application errors.

Developers on the artistic end of the spectrum, however, struggled with debugging. This aligns with results from Papier Mache, where they found debugging to be the hardest part of ubiquitous computing application development. They observed developers becoming confused by sensing errors that were "mysterious" from their perspective (Klemmer, et al., 2004). ARTIST, RESEARCHER, DIRECTOR, and DEVELOPER all recounted their struggles with, not just eliminating errors, but with first *isolating the cause of them*. They described their confusion when working with the tracking system and the workarounds they developed when the cause of the errors could not be found. For example, they put constant offset variables in their applications

thinking DART was causing a static tracking error, when in fact the head tracker was mounted incorrectly on the HMD. DEVELOPER explained that they modified their designs for Woyzek based on what they could and could not get working with the IS900 tracker and struggled with calibrating the position based triggers. It is impossible to know if they were encountering failures of the tracking hardware, bugs within DART, or if they were configuring their application incorrectly.

These anecdotes highlighted DART's lack of support for debugging the larger system *outside of the software*, the "quagmire of tediousness" referred to by Greenberg (Greenberg & Fitchett, 2001). We did create components that helped the developers inspect their 3D scene and tracking data. For example, a 2D overhead map widget made it possible to get a "World in Miniature" (Stafford & Piekarski, 2008) view of the scene and assisted developers with a common complaint of not being able to find their augmentations in the world. Capture/playback features allowed developers to carefully recreate a tracking situation or revisit anomalous behavior via recorded data. However, this assumed that elements outside of the DART environment were working correctly. Therefore, when hardware, such as cameras or trackers, was not behaving as expected or when 3D models imported incorrectly there was less high-level trouble-shooting support; developers were expected to devise their own tests via scripts, refer to debugging statements we sent to the console, or investigate via external programs for those systems (e.g., configuration software that came with the tracker, a camera utility program, or the 3D exporter in their modeling package).

The DART components were built with a *computational thinker's debugging strategy* in mind, like those exhibited by TECH, who knew how to break down the system into functional elements and how to investigate errors in a systematic way. Perhaps "always on" features like the animated commands in the VR authoring tool, Alice, would have been more appropriate (Conway, et al., 2000). This feature showed the visual effect of every command as it was entered, helping developers see the results of

their choices immediately and helping to reveal the cause of errors. It is clear that an area of future investigation relates to creating AR debugging constructs that are appropriate for a designer's expertise and development approach.

Rapid Prototyping Support

The interviews revealed that DART was used for prototyping. NOVICE, ARTIST, and RESEARCHER indicated that DART was often the *first step in an AR project*, regardless of whether it was used for full deployment or not. ARTIST and RESEARCHER commented that recently students have created early versions of experiences in DART before implementing them fully in a different tool (e.g. an early version of "52 Card Psycho" (Rothenstein & Sizintsev, 2008) Figure 27. (c) was prototyped in DART before being implemented in SnapDragonAR). NOVICE commented that even now he still occasionally uses DART to rapidly demonstrate AR concepts to potential clients. ARTIST commented that, "DART was forgiving. Letting people start with baby steps." A review of the DART mailing list and previous responses to a DART survey show that developers found DART appropriate for *early explorations of an idea*, which could then lead to a full implementation in DART. This suggests that we met our goal of DART supporting the *full range of the development cycle*, beginning with brainstorming and prototyping.

The concept of providing methods for simulating sensor input to aid in prototyping, similar to features of DART, were present in other toolkits including Phidgets and the Context Toolkit (Dey, et al., 2001; Greenberg & Fitchett, 2001). However, similar DART components designed to support rapid prototyping were under-utilized by external developers. Internally, we used these components extensively. As discussed in Chapter 5, we used capture/playback for prototyping exercises in courses, for debugging/testing, and to easily capture animations and video for use in the final version of the application. Wizard of Oz features were utilized in the Voices of Oakland

project for evaluation as well as a replacement for GPS in the deployed version. However, we have few reports of external developers realizing the potential of these components in their projects. The interviews suggest that the developers were not used to having this functionality in an authoring tool; so they chose to develop with traditional approaches. All of the interviewees made positive comments about the concept of capture/playback. MUSICIAN did not use the DART version, but he said that he had attempted to include the concept in his own tools. NOVICE used it for testing of "Apollo and Beyond." DIRECTOR commented that they considered using the capture capabilities to create a personalized video of each visitor's experience, but did not have time to implement the idea. It appears that external developers were even less likely to utilize the WoZ components. ARTIST admitted that she did not realize WoZ features were actual tangible components in DART. The only interviewee that worked with the WoZ features was NOVICE, who used it for networking between his DART applications.

The question is, why did the external developers not make greater use of these features? The interviews reveal a *lack of awareness* of these components and an impression that *too much time* would be required to learn how to use them. They recognized that using these features could have benefits, but they felt that in their own projects it was more efficient to use traditional methods of development and testing. They made the decision many developers make, which is to forge ahead with development rather than spending time on something new that might not provide value. As the Designer's Outpost team found, it is necessary to *minimize extra effort* required by users in order for them to integrate a new tool into their workflow (Klemmer, Newman, Farrell, Meza, & Landay, 2000).

This issue dove tails with the previous discussion of debugging needs. It is clear developers, particularly those less experienced with computational thinking, struggled with debugging their applications. I believe we did not do an adequate job of demonstrating the value of these components to our community and did not provide

enough support to help the developers learn how to use them. We made assumptions that developers would immediately *understand the potential* of these features, but our mistake was assuming the developers would have experience with and an advanced understanding of software debugging techniques. DART was informed heavily by rapid prototyping research (Klemmer, et al., 2004; Klemmer, et al., 2001; Klemmer, Sinha, et al., 2000; Landay & Myers, 2001; Li, et al., 2007; Li, Hong, & Landay, 2004). We assumed that our users would be aware of this research (e.g. the value of informal content for early experience testing and the potential of transforming early ideation artifacts into technology prototypes) and that they would be familiar with the unique challenges of AR authoring (e.g. the need for in situ prototyping and the challenges of debugging). Therefore, we provided low-level examples of how to use capture/playback and WoZ components (see Appendix G), thinking that developers would only need guidance in syntax, but our target users were, by design, those who had not previously worked with AR and, perhaps, had little large-scale software development experience. Upon reflection it is clear they required guidance regarding the *AR development process in general* and that we should have also disseminated *sophisticated example projects* to illustrate the value of the features and to provide reusable templates for common uses.

Community

One reason for the wide adoption of the ARToolkit was the active developer community. The mailing list ("ARToolkit Mailing List Archive," 2000), which was active for a decade, was often the resource used by novices to help them overcome the initial learning curve. A perusal of the archive shows users struggling with common problems (e.g., “Newbie Question: error LNK2001”, “loading VRML objects into ARToolkit!”, “anyone with idea about the camera”) and experts, including the development team, quickly responding with advice. A lesson learned via my interviews was the *importance of community* to the success of DART as well. We originally

assumed that the greater Director community would provide valuable resources to those learning and working with DART. However, as discussed in previous sections, the popularity of Director was in decline even when DART was first released. The interviewees and posts to the mailing list revealed that many attributed their success with DART to the mailing list and the availability of the development team to answer questions. NOVICE's comment "if you guys hadn't had that mailing list we wouldn't have gotten off the ground," echoed the sentiment of other developers who explained that support to get them over initial problems (e.g. getting DART to access their camera, errors with 3D model loading, and a blank screen caused by Director not using the OpenGL renderer) was critical to their projects.

During the interview ARTIST thanked me for all the support we provided her as she built her AR research lab and remarked that we were a "model of intellectual generosity and curiosity." She commented that they could have even used more help in learning the full capabilities of DART, but they were concerned that they might "exhaust [our] goodwill." The importance of *responsive technical support* to the interviewees' projects and the level to which they cited the mailing list as an invaluable resource was surprising. I also theorize that our responsiveness to user questions and an active online community gave the project *a sense of permanence*. Perhaps these were signals to authors that this was an active project that would not suddenly go dormant. The lesson to be learned is that the value of these low-tech components of an authoring tool ecosystem should not be overlooked.

However, feedback also shows us that the community would have benefited from *more bundled examples and documentation*. We believed that a combination of the script source coupled with an exhaustive set of small examples would be sufficient. Each script (i.e., feature) in DART had an associated "template app" that we included with the application download (see Appendices F and G). These template apps consisted of a small Director project that showed the capabilities of each component and an

accompanying text file that explained the meaning and value range of each parameter in the script, stated how the application worked, provided instructions on how to run the application and described what the user would see as a result. We also provided two high level documents, one that was an overall description of the DART architecture (see Appendix C) and another that was a collection of answers to frequently asked questions and authoring advice we generated over time (see Appendix E).

However, the interviews revealed that this level of documentation was not sufficient. DEVELOPER did state that she relied on these resources to learn DART and found them helpful, but others made comments about the lack of documentation. It is not surprising that those most affected by the lack of rich documentation were the less technologically expert developers. MUSICIAN explained that his DART explorations never advanced to a full application due to the dearth of sample applications. ARTIST, RESEARCHER, and DIRECTOR all commented that they would have preferred much more complex applications to refer to rather than the small templates. These complex projects would have shown the full capabilities of DART (e.g., DIRECTOR commented that she knew her vision for Woyzek was possible because of having seen FAM) and could have *established design patterns* for common features. This might have helped with the adoption of advanced prototyping features discussed in the previous section.

RENAISSANCE explained that he was less interested in "nuts and bolts" examples of how to do things in DART and more interested in prototypes that illustrated *best practices and successful AR interfaces* (i.e., less of the "how" to accomplish things in DART and more of "what" to build in AR). As with the prototyping features discussed previously, it appears that a mistake the DART team made was in assuming all our users would have a firm grasp of the capabilities and limitations of AR and that they would be experienced building an understanding of a development tool based on source code and sparse functional examples.

**Reflections on AR Authoring**

DART was a tool that allowed us to study and explore the AR authoring process. I was interested to collect reflections from the interviewees regarding the challenges of AR authoring and on what is needed in an ideal workflow or tool informed by their experiences with DART. The non-DART AR experts (e.g. PROFESSOR, STUDENT, and MANAGER) provided additional insights. The reflections fall into the categories of *access*, *affordances and constraints*, *layered authoring*, *prototyping*, *collaboration with diverse teams*, and *beyond the software*.

Access

ARTIST commented on a continuing "*hunger for tools*" among AR designers. A common theme amongst all of my interviewees was a feeling that, while there are far more tools for AR authoring than there were 10 years ago, these tools support applications that are less sophisticated in many ways than those supported by DART. The needs of these developers differ from those who are focused on building commercial applications to be deployed on smart phones and game consoles. This is a population of researchers and artists interested in exploring the future of AR, technologically and creatively. They are building applications that use *esoteric hardware*, are created for *carefully designed physical spaces*, utilize large amounts of *high quality content*, rely on *new types of users interactions*, and leverage a *significant connection between physical and virtual objects*. And yet, most of them are not technologists. They want to focus their efforts on design rather than on developing their own tool chain or working at a very low level with hardware and software.

DART and its modern precursors have much in common with standard UI toolkits. The advent of toolkits, such at the Macintosh Toolbox (Computers, 1992), helped users quickly develop with what became the standard building block of GUIs, scrollbars, dialogs, and buttons, without extensive programming. These toolkits caused an

explosion of GUI applications with a consistent interface. However, these tools also serve as a disincentive to explore other types of interfaces that were not supported by the toolkit, such as two-handed input (Latulipe, Mann, Kaplan, & Clarke, 2006). Similarly, modern AR tools, such as Vuforia in Unity3D, are approachable, yet their architecture *guides users to create specific types of AR experiences* (e.g., 3D virtual worlds rendered on top of printed markers). The layered authoring, physical objects as first class components, and support for arbitrary hardware in DART was our attempt to encourage authors to create a wide range of MR applications. I argue that the array of applications highlighted in this thesis are diverse, ranging from TUIs on tabletops, to immersive installations using HMDs, to 3D "toys" controlled by printed markers. However, the design of DART certainly did have an impact on the type of applications that were created. Many of our DART features were driven by application needs of early applications like FAM. It was this influence from FAM that led us to create full-featured support for video content, which was then used heavily by our external developers.

Every interviewee who had previously worked with non-vision based tracking hardware, HMDs, or custom interaction devices (e.g. TECH, NOVICE, ARTIST, RESEARCHER, DIRECTOR, DEVELOPER) commented that current tools provide few, if any, features to support working with these components. However, they noted a need for access beyond just that of hardware. In general the less technologically expert developers expressed a need for access to AR in general. They noted the need for a "lower barrier to entry."

PROFESSOR explained that for his students and the types of experiences they are creating, authoring "*doesn't have to be easy. It just has to be possible*." Artists are familiar with working this way, developing a workflow, even if it is tedious, and then focusing on exploring and refining a creative concept. For example, in his AR design courses, the students are focused on creating mobile AR applications with Argon (see Chapter 7) that leverage background panoramas rather than live video. This is a simple

concept technology-wise that is under-explored creatively. He commented that his technology focused collaborators in computer science concentrate on increasingly novel and sophisticated features for AR applications. The role of a humanist, such as himself, is to instead advance the creative boundaries of a medium. Authoring tools are needed that support this type of *creative exploration*.

Affordances and Constraints

PROFESSOR also commented that computer science students doing AR projects focus on trying to solve technical hurdles. Conversely, his digital media students want to focus on creative exploration, but their work is hindered by a *lack of technical understanding*; as a result they envision novel and interesting concepts that are unrealistic to implement. He and the other design oriented interviewees shared anecdotes from their own or student projects where either creativity was stifled because the resulting application did not take full advantage of the technology (e.g. ARTIST described how film students would focus on single POV videos because they did not understand the affordances of AR) or where frustration arose when envisioned designs were technologically impossible (e.g. DIRECTOR abandoned ideas for "Woyzek" when the DEVELOPER did not know how to implement them).

PROFESSOR also described a phenomenon he commonly observes, which is his students *authoring in a way they understand* that is also "wrong." He has seen that this can have negative effects on the experience (e.g. slow performance, unexpected behavior, non-ideal interfaces etc.) and the student developers do not understand that their approach is to blame. Therefore, there is a tension between letting them author in whatever way works for their mental model and process, since "beautiful software engineering is not always required", and keeping them from "creating a subpar artifact through this hackery." Similar phenomena were observed in the Papier Mache project where the novel hardware and computer vision required for TUIs meant that developers did not fully

understand the constraints of the system. They described a developer who had "the lingering impression that the system must be broken, when in fact the system was just being slow because [they] were pushing the limits of computation speed" (Klemmer, et al., 2004).

RENAISSANCE pointed out that, in his opinion, it is important for the authoring environment to support the developer in *designing appropriate AR interfaces*. In his opinion AR applications can "feel clunky and gimmicky." Whereas, his primary goal as an artists and researcher was to make experiences that felt *natural and intrinsically compelling*, "not just cool for being an AR project." He stated, "We want to tap into the unique affordances of AR. The biggest challenges with AR then are what they always are - how to DESIGN AROUND THE AFFORDANCES [emphasis his] in a clever and creative way. I think of AR development like learning some obscure musical instrument that has all these strange things to watch out for (like registration, lag -- on phones, etc.)." He argued strongly that any authoring tool should *reveal technical constraints*, helping developers *understand their impact*, how to *design around them*, and guide developers toward *appropriate interaction choices*. He also commented that integrating such knowledge or heuristics into a tool is challenging because these constraints and affordances are constantly changing.

STUDENT commented that providing this level of guidance with an authoring tool is also challenging at this point in time due to a lack of conventions both for AR interfaces and for authoring paradigms. He queried, "Could you even build the "Final Cut" of AR now?" Others in the AR community are tackling this issue by carrying out HCI studies of applications (Schmalstieg & Wagner, 2007; Wagner et al., 2009) and interactions (Henderson & Feiner, 2011; Oda & Feiner, 2009), while others are developing design patterns for application domains such as gaming (Yan et al., 2011). This research will inform future authoring ecosystems. These future systems should not only consist of a software authoring environment that assists in the design of appropriate

effective applications, but must also provide a community and libraries of shared projects that will help developers realize what is possible, which is often the key to innovation.

Layered Authoring

Regardless of technical proficiency, interviewees consistently expressed a desire for a tool to support *early prototyping without coding*. TECH described his desire for a system that let him tag physical objects or locations for later content placement. ARTIST expressed her desire to identify physical locations by moving through the space and/or interacting with a 2D map of the space. RESEARCHER created a custom authoring tool to allow his students to link video and markers quickly, with no programming. However, this does not mean that the solution is categorically an authoring tool that does not require scripting. Such tools can be useful, but typically they support a limited range of applications (e.g. Catomir (Zauner & Haller, 2004), AR authoring in Quartz Composer (Cameron, 2012), and Farrago (Warne & Wozniewski, 2011)). However, simple authoring interfaces can be more powerful when they are part of a layered authoring environment that allows developers to engage at the desired level of complexity, where lower level access requires greater technical sophistication but also allows for greater flexibility.

Other toolkits have provided *layered authoring*. For example, ComposAR allows the developer to author with direct manipulation and tangible interactions or extend the tool via Python (Seichter, et al., 2008). And I believe that the applications demonstrated by my group of interviewees illustrate the value of layered authoring tools in DART. The projects described in this chapter represent dramatically different types of applications (e.g. user interface experiments, tangible user interfaces, art installations) with different technological, content, and interface requirements. The layered authoring in DART made it possible for TECH to add support at a low level for shared video streaming, for NOVICE to create a low-cost tabletop tangible user interface and to implement a

networked application despite his lack of experience in the area, while film students were able to create artistic AR installations without any previous programming experience. The layered approach supports a range of developer abilities and foci. This is of particular import on large projects such as "Woyzek" where some contributors barely interacted with the AR aspects of the project while others handle technically sophisticated parts, and yet effective and efficient collaboration was critical.

STUDENT described an art installation project in which he was the technical lead, built for a mobile platform using C++ tools for development (Barba, Rouse, Bolter, & MacIntyre, 2010). The installation was situated inside of a shipping container and invited the users to explore the space with handheld devices and to interact with props, learning about the history and global impact of shipping containers. He discussed the workflow he and his non-technical collaborator developed during the project (to be discussed more thoroughly in the "Collaboration with Diverse Teams" section in this chapter). Germane to the discussion of layered authoring, he wished there had been a mechanism to allow the designer to participate in lightweight technical tasks such as loading and testing of content which would have freed up for time for him to focus on programming the logic and interactivity. However, their tools, which were exclusively low-level, were totally inaccessible to the collaborator even for simple tasks. When asked what the ideal tool would have been for their project, he jokingly responded "a drag and drop tool for creating shipping container art installations." However, this comment points to a possible future direction for layered authoring tools. Perhaps what are needed are *meta-tools* that let teams begin a project by coopting or building their own simple authoring environment. Like the other interviewees, STUDENT commented that he needed a simple environment to "*help with physical layout…make spatial relationships easy*." This concept has been previously explored in domains such as ubiquitous computing (Li, et al., 2004) and is currently a topic among those creating web-based AR tools (see Chapter 7).

Prototyping

Prototyping has already been discussed extensively in this chapter, as it is these early explorations that are often critical to the outcome of the project. ARTIST and RESEARCH believe that they were never able to achieve the "critical mass" of AR applications needed to fully meet their initial research goals because students were not able to understand the potential of AR and to become proficient with DART fast enough. As a result they were not able to iterate on their ideas as many times as was needed to fully explore the design space. Meanwhile, NOVICE and his collaborator were energized by their early design explorations with DART and this inspired them to pursue their ambitious project. RENAISSANCE described his desire for an "early ideation sketchpad" for AR. He has developed a workflow for Unity3D that "affords a tight iterative cycle of thinking and implementing and testing and improving" that is integral to his research.

The interviews revealed that developers approach brainstorming and prototyping from two different directions depending on their expertise and project focus. One group, consisting of those with more advanced technical abilities (e.g. TECH, NOVICE, STUDENT, DEVELOPER, RESEARCHER, and MANAGER), approach from the *technology side* of the project. They try out technical scenarios to ensure that they will work as predicted. They mock-up ideas using the authoring tool and share those with their designer collaborators to convey their ideas. At the other end of the spectrum, developers, such as DIRECTOR and ARTIST's students, may initially avoid technology and *focus on user experience design*. Their tools are sketches, videos, transparencies, and physical models.

STUDENT's description of the prototyping process for the InBox project exhibited both aspects of prototyping, a traditional approach guided by his non-technologist collaborator, and a technologically enhanced approach led by him. The beginning of the process was influenced by his collaborator's experience from working in theater. When they began the project neither had ever even set foot inside of a shipping

container, so they began by looking at pictures of them online and reading books about their history. They drew sketches on transparencies to brainstorm. Then they made a mockup of the shipping container space with tape on the floor to indicate locations of props and large rolling whiteboards to form the “walls” of the container, and then put printed imagery up on these walls. This allowed them to experiment with the “set design” of the space. STUDENT commented that during this staging process they would notice things like "oh, that will block the door, or that will block the projector, move it over here.” Later this same space was used for technology tests as well. STUDENT would preview the AR software there and they also ran a pilot user evaluation in the space. However, for day-to-day development STUDENT needed an easily accessible test environment. He explained, "I remember I had a collection of frame markers on my desk and I had super miniaturized everything." This miniature test environment allowed him to easily analyze the application logic and interactions. His anecdotes illustrate the *important role that physical space and objects play* in the design of certain AR experiences as well as the reliance that many non-technologists have on traditional design techniques from their domain (e.g. theater, film, music performance, etc.). Unfortunately, the low level tools being used for the AR development in this project were disconnected from all the prototyping and design efforts. For example, there was no support for quickly translating decisions made with the tangible items in the mocked-up container into software, nor did the tools provide any rapid way for STUDENT to create the miniaturized version of the system for testing.

Over the years we added components to DART, such as the sketch interface for storyboarding (Presti, et al., 2005), to attempt to support *transitions from these low, or no, tech products to an AR application*; features such as the flexible tracking architecture (Gandy, et al., 2004) allowed the developer to rapidly create versions of the application that were portable, at a different scale, or disconnected from live trackers which were onerous to access for testing. However, these were small steps toward the goal of

seamless and efficient prototyping. And while we learned from DART that it is important for a tool to support rapid prototyping, I also do not believe that the only answer is a single all-encompassing authoring tool. Rather, what are needed are components of technology "glue" that allow designers to utilize their *existing workflows and tools* in design scenarios such as InBox. "Designer's Outpost" shows an example of this approach. Their users expressed a preference for pen and paper for early ideation. The researchers envisioned Designer's Outpost as part of an ecosystem of tools that supported a fluid migration of low-tech artifacts to technology prototypes (Klemmer, et al., 2001). Such components address the current shortcoming of traditional tangible methods, which is that typically these assets cannot evolve into an AR application.

Collaboration with Diverse Teams

Projects such as "Woyzek" show that diverse teams are often required for ambitious AR projects. I discussed the issues related to collaboration with my interviewees. There was an agreement among many of them that, as AR projects become more sophisticated and commercial, we will see a *movement toward specialization* among contributors, a process that plays out in other media such as film. When developing the requirements for AR authoring tools, then, it is necessary to consider to what extent each contributor will understand and participate in all the aspects of the AR design and development process. The interviewees presented contrasting views on the roles of AR collaborators. For example DIRECTOR commented that she prefers to "focus on making art" and is not interested in developing advanced programming expertise, while others feel that AR authoring requires all contributors to possess a wide range of expertise, including the technical. STUDENT shared his opinion that it is important for the workflow to support a hierarchy of collaborators that includes a project leader who understands all aspects of the development process, both technical and creative.

PROFESSOR described his view on diverse teams in the context of film. He pointed out that screenwriters do not engage with technology and they are not typically special effects experts, but they have clearly internalized the capabilities of film such that they can write scenes appropriate for the technology and budget of the film. He also commented that it is "built into their DNA what movies look like." On traditional mainstream movies the roles are very defined and compartmentalized, “everyone involved from the sound designers to the costumer knows how the medium works.”

However, in the case of movies that push the technological or creative envelope (e.g. “Sin City”), the boundaries between roles such as writing, camera control, set design, and special effects become blurred and greater collaboration is required. For example, in Sin City the production team created technology tests for difficult shots that were stylistically important (e.g. people in shadow with their eyes illuminated and white glowing blood on characters). To achieve these effects they had to developed creative techniques that involved makeup, props, and costumes that were carefully chosen (Rodriguez, 2005). I believe that AR is still in an early stage that requires a similar approach to projects. We lack standard design and development approaches. Few people working on projects have AR “built into their DNA” yet. Diverse people working on AR projects must be more like the “Sin City” team, where *all participants engage with technology and participate in exploring the medium together*. This is meaningful when considering AR authoring needs. The AR medium is not yet ready for the highly compartmentalized well-defined process that takes place in mainstream filmmaking or video game development. Therefore, the work flow, which includes AR specific authoring tools as well as established media software such as Adobe Photoshop, must consider the expertise and goals of various contributors, but should also allow for a blurring of the boundaries between them (e.g. developer, designer, choreographer, camera operator etc.)

ARTIST and RESEARCHER expressed similar views. They emphasized that there is a difference in what they expect from their students in an educational setting versus what expertise would be required in practice. They feel that the students must engage directly with the technology and gain experience with all aspects of AR authoring in order to create that deep understanding of the medium referenced by PROFESSOR. Therefore, authoring tools for use in AR education must allow students to gain this experience even if they do not typically work closely with technology.

MUSICIAN describes his experience as a technical novice attempting to work with technical collaborators to create his AR music experiences. Originally they were using ARToolkit and, therefore, it was not possible for MUSICIAN to contribute directly to the technical components of the project. He explained, “It was frustrating to try to explain what I wanted to the programmers (in Japanese!), especially if I didn't completely know yet what was needed” and "the feeling was like that of a painter who was not allowed to touch the paintbrush but, rather, had to relay instructions to someone else who would then go to another room, apply the paint, then bring the work-in-progress back for the painter to see and suggest corrections. This breaks the traditional dialogue with one's materials that is an essential part of the creative process." Later he utilized Pure Data, a real-time graphical programming environment for audio, video, and graphical processing, to prototype the sound part on his own, prior to engaging with the developers. He commented, “the attraction of tools like DART was that I could actually do it myself and stay up all night if necessary rather than keep some poor programmer up all night then change my mind the next day." This resonates with PROFESSOR’s comments in the prototyping section regarding the need for artists to simply have tools that let them explore and iterate on ideas; even if the workflow is tedious, this is outweighed by the value of access.

MANAGER’s anecdote provides the most diverse collaboration example of all the interviewees. He led a team of computer scientists and 3D artists who collaborated

with the rock band, Duran Duran, to develop an AR system for use during their live shows (See Chapter 3). MANAGER explained that the challenges, as well as the value, of working with a diverse set of stakeholders were highlighted once the technology was ready to preview in a real stage setting. He and his team did a live rehearsal before the first show to test what they could and could not do with the band and crew during a show. MANAGER explained that this process was extremely frustrating because, for awhile, it seemed as though the concepts would be rejected by the tour manager and other people involved who did not want to use any of the effects. There was a degree of politics that had to be worked out and comfort that had to be built. A seasoned crew does not like to add unknown components to a show since it can make their jobs more complicated. MANAGER commented that communication was key. He made a point to ensure that as many people in the band and the crew as possible were part of the creative development process. He explained, "I wanted everyone in the band and the crew to feel a sense of ownership in the project." Ultimately, the projectionist and video designer/editor provided significant help with the system design. By the end of the tour the crew was impressed with the AR experience, but they had to first be convinced that that technology would work reliably and would add value.

Regarding how the effects were used in the show, the process evolved even as the tour progressed. MANAGER explained, "By the time we went to Birmingham, their hometown, they really understood what you could and could not do. We came up with some great ideas with the band and used them in that show. It was the best one."

Based on the comments from the interviewees it was clear that a significant challenge in creating AR applications with diverse teams is the need to *convey ideas and technical constraints to other contributors so that they may participate effectively*. In FAM, Woyzek, the Duran Duran project, and InBox there was a critical initial step, which was helping the non-technical collaborators to fully understand the affordances and technical constraints of AR. These collaborators range from the actors in FAM, to the

Woyzek choreographer, to the projectionist for Duran Duran. It was important to build their enthusiasm for the project and to get to the point where they could actively contribute to the design and development process. As MANAGER described, the Duran Duran system was improved in an iterative process due to the contributions of the other stakeholders. An authoring ecosystem for this type of project must accelerate and improve this process.

Beyond the Software

The interviews revealed that many developers such as DIRECTOR, NOVICE, STUDENT, and MANAGER felt overwhelmed and exhausted by their ambitious AR projects. For example, both NOVICE and STUDENT temporarily vowed to never work with AR again at the end of their projects. It is these types of applications that fully explore the potential of AR, however the work required a diverse set of expertise that caused the interviewees to feel frustrated and fatigued. A commonality was that these projects required significant tangible design and construction of space, objects, and electronics.

NOVICE kept a blog tracing the development of "Apollo and Beyond" (Barker, 2006a). It detailed their confusion with choosing appropriate cameras, understanding lens characteristics and calibration. They also struggled with configuring a wireless video system for the handheld viewers. Creating the tangible projection table was challenging, as they had to use trial-and-error to determine how to mount the projector and camera such that the camera could see the marker cubes and the projection would cover the table. NOVICE described, "the ceiling wasn't high enough so we had to use a mirror, figure out how to remove keystoning and determine how far away the camera could be. It was trigonometry and optics." They had to investigate topics like materials for the mirror, how thick the mirror had to be to keep it from bowing under its own weight and "a lot of things we never thought we'd have to deal with."

In DIRECTOR's "making of" documentary for "Woyzek" we see her constructing a stage, experimenting with stage lighting, directing the dancers, and working on musical lyrics with the composer (Quinsland, 2007). She commented in the interview "*every step, there was so much more to learn and so much to do*."

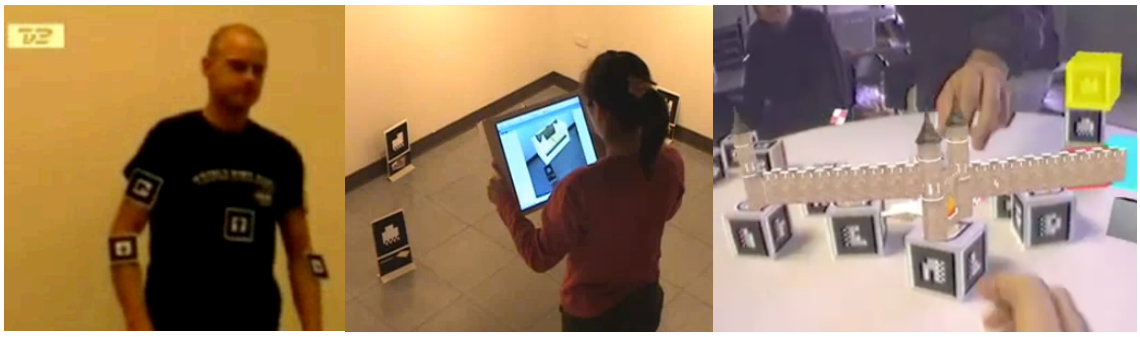
Some of these struggles are inherent to large ambitious projects and are not that different from the challenges of staging a play, shooting a film, or building a video game. However, I think it is valuable for those researching AR design and authoring tools to consider these elements of the process that exist outside the bounds of the software. While adding additional features to an authoring environment could have relieved some of the aforementioned struggles (e.g. utilities to help novices determine camera specifications, hardware prototyping support, or pre-visualization support for those filming augmentation videos) some low-tech resources such as approachable guides on camera calibration might have helped as well. In other more established media such as film there are workflows and a vast array of resources, software, and equipment designed to make production as efficient as possible. As AR matures we must craft similar techniques and workflows to make it less painful to create large-scale AR experiences.

## Summary

We know that DART was used for many projects ranging from prototypes and demonstrations to real systems that were experienced by the public. However, it was challenging to rigorously evaluate DART since it was a tool we released freely into the community without a formal mechanism, beyond the mailing list, for tracking the developers and its use. A shortcoming of this qualitative study is that it was performed years after the final release of DART. Therefore, it was challenging to locate and communicate with DART developers. As a result, there are only eight participants and their areas of expertise are very diverse. Also, the recruitment strategy, of finding developers that we knew to have built DART systems, meant that my results reflect the

experiences of those people who found DART usable and useful. Only MUSICIAN had a less positive experience with DART and thus did not use it for a full project. There are undoubtedly many others that got no value from DART, potential users who downloaded it, tried it, and found it lacking. Unfortunately, it was difficult to identify and locate such people years later. However, the amount of time since DART's release also provides benefits to this study. For example, ARTIST & RESEARCHER used the tool for years in a variety of ways. They were able to comment on DART from the perspective of having used it for a long period of time in a vast number of projects and from building their own DART-influenced tools. All the participants were able to provide feedback on DART, and AR authoring in general, in the context of other modern tools. I believe the participants' feedback on DART has more significance since it is tempered by the passage of time and the advancement of the AR medium.

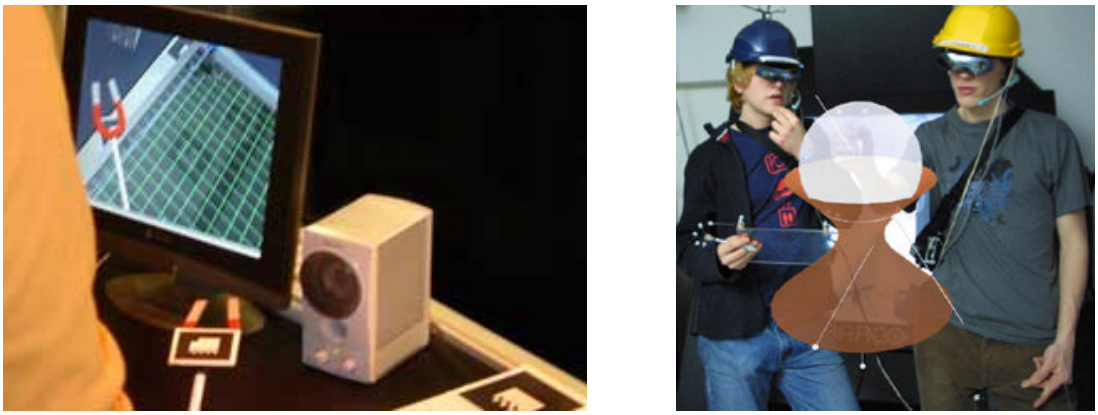
Anecdotal evidence suggests that DART was used for a variety of intriguing applications. During my research on external projects built with DART, I literally heard rumors of projects built with DART that I could not verify (e.g. RESEARCHER said, "I had a colleague who used [DART] for city planning." And ARTIST described a conversation she had at SIGGRAPH 2006 with someone that built a commercial haunted house installation in Belgium with DART). Exploration on YouTube uncovered videos of DART projects (see Figure 31. ), while web searches revealed sites describing applications built with DART (see Figure 32. ). During my research, I also found a book written on using DART, Blender, and ARToolkit for architectural visualization (Hohl, 2008). However, it is challenging to fully quantify DART's impact. I believe the evidence gathered from interviewees, the mailing list, face-to-face conversations, and web searches shows that DART was used by authors with a *wide range of expertise and goals* to successfully create many AR experiences.



(a) (b) (c)

**Figure 31.** Screenshots from YouTube videos of DART projects (a) a demonstration of "ARDressCode", an application for virtually trying on clothes, on the television show "Guten Morgan Denmark" (b) a woman uses "AR—My Place" to preview furniture in a room my place (echopai, 2007) (c) an AR project from Utrecht School of Arts (punksmurfjie, 2006)

Our target users for DART were technical designers, like those we had worked with on TAM, design focused people who were experienced with "computational thinking" even if they would not consider themselves software developers. The study of DART use showed that such authors, like RENAISSANCE and NOVICE, were able to create very computationally sophisticated applications quickly, "*lower the floor*" for development. For example, TECH admitted that he was capable of building his experimental systems in C++ and the ARToolkit, yet he chose DART because he knew it would allow him to accomplish his project goals quicker and more easily. However, it is notable that DART also "*raised the ceiling*" for non-computational thinkers such as ARTIST's students and DIRECTOR. These users stated that they would not have been able to explore the medium extensively, or possibly at all, if they had not had DART.



(a) (b)

**Figure 32.** (a) Robotract, a tangible augmented reality game (Olden, Bruin, & Kousemaker, 2004) (b) torus surfaces of descriptive geometry in augmented reality (Lima, Cunha, & Haguenauer, 2009).

DART provided an architecture that made it possible to create a wide variety of AR applications. The generic DART components that represented abstractions of core AR elements and common application features allowed developers to quickly build powerful systems with simple scripting. The lessons learned from DART directly informed our own subsequent tools (see Chapter 7) as well as Qualcomm's Vuforia.

The experiences of the interviewees *validated our AR authoring guidelines*. They utilized DART for all four stages of the AR design process. They created rapid prototypes and designed applications where the physical world was a key component of the experience. Based on their feedback, one of most useful components of DART was the flexible and powerful hardware access. They successfully collaborated with diverse teams to create their systems and took advantage of the layered access provided by DART. In their interviews the developers discussed the importance of supporting ideation and allowing those artifacts to drive early versions of the application. They also described the challenges of debugging and provided insight into how future tools could address them. They acknowledged the significant role that content pipeline plays, echoing the

need to iteratively create and test with proxy/prototype content, praising the power of Video Actors in DART and complaining about their struggles with importing 3D models.

The interviewees also provided some insights that were not captured in the original guidelines. Their feedback showed us that *reuse of project components* is of critical importance and that many high level tools are overlooking this need. They emphasized the critical role that *community* played in their experience with DART. They highlighted the need for different types of documentation and *example applications*, pointing out that often they needed direction in "what" to do with AR rather than just "how." They wanted guidance from the authoring tool that helps them properly design for the *affordances and constraints* of AR. Lastly, they need help understanding and *debugging all the components* of their AR system, even those, which were outside the bounds of their DART application.

The interviewees expressed a desire for *meta-tools* that would allow them to create very simple application specific authoring utilities. Modern web technologies may form the basis of such tools. As I will discuss in the next chapter, Argon is already being used to author specific high-level tools for domains such as tour creation (Su & Feng, 2012). It may be that the future of authoring is more than rapid prototyping, but rapid prototyping of the authoring tools themselves.

Feedback from interviewees, as well as our own experiences, showed that current AR tools are overlooking some of the AR specific authoring needs. For example, they need to support a greater range of hardware access. In general, there is a need for more recognition by such tools that the physical world is a critical component of an AR application, from brainstorming, to debugging and deployment. The interviews revealed the extent to which developers consider, design around, and work in the physical world or a simulacra of it such as STUDENT's mock-up shipping container, and yet most current authoring tools are mainly focused on tracking and the placement of final versions of virtual content.

# CHAPTER 7

# TRACING IMPACT

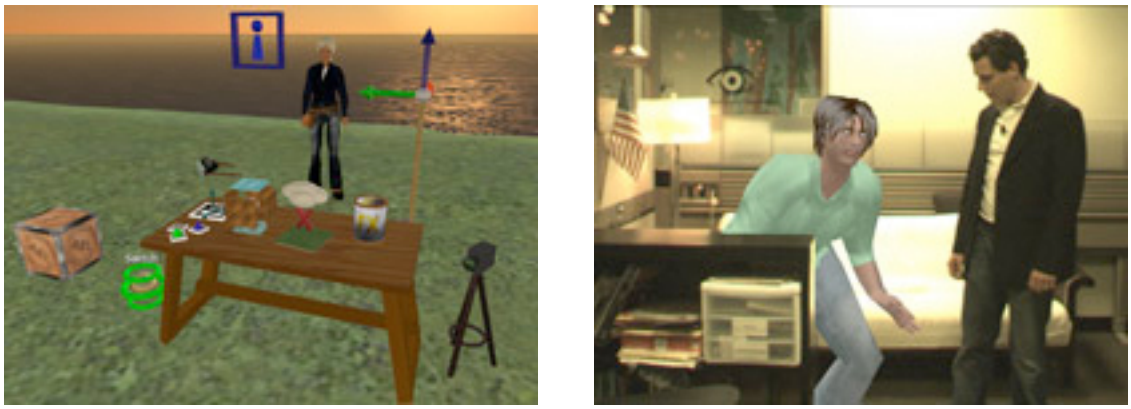
My experiences with collaborative AR projects including artists, designers, psychologists, performers, and media theorists informed a series of subsequent authoring tools, as well as a culture in our research group (AEL) of working in diverse teams and sharing responsibilities. In our projects we avoid the tradition of sharply delineated roles with different stakeholders only focusing on their area of expertise, "tossing" output "over the wall" to each other. Our overall goal is to make it possible for people who are not AR researchers to explore the AR application space. It is only through participation of these types of developers that the promise of AR can be realized. Just as the web saw a rapid growth in creativity, utility, and sophistication when tools made it possible for a very diverse and vast population of people to develop websites, we believe that AR needs this infusion of different perspectives and expertise to mature.

The work that went into identifying AR authoring guidelines, the development of DART, and the applications and collaborations that resulted from it continue to influence research and tools in the AEL, showing the value of the first three contributions of this dissertation. While others lead these projects, they are in the DART tradition. In this chapter I trace two main threads to highlight impact: the development of the *AR Second Life system*, and the creation of the *Argon AR web browser*. Both initiatives are examples of our growing need to facilitate broader and more realistic systems via useful tools. Also, as discussed in the previous chapter, often there is a tension between the "right" way to develop AR applications (e.g. the most accurate registration, the most realistic visuals etc.) and what meshes best with how people outside of this domain work. In our DART research we erred on the side of providing tools that fit with existing workflows and/or design goals and this is true for both ARSL and Argon as well.

## AR Second Life

As an AR authoring tool DART was effective. However, over time multiple issues arose that required a new generation of authoring tools in the AEL. At the time we started the DART project, and for the next few years, Director was a widely used interactive media design tool, however, its utility waned. From our experiences using DART in our own projects, utilizing it in courses, and feedback from our external developer community we found that its programming model, proprietary scripting language, and closed 3D engine could appear complicated and foreign to new users from outside the interactive media domain. For example, we found that many of our computing students were unmotivated to become Director experts, especially as Director became less popular over the years. In the meantime a new generation of media development tools and game engines had emerged. At the same time, in AR research, a new class of problems come to the fore, which were content and experience heavy but no longer required sophisticated programming to realize them.

There was also an increasing interest in leveraging network/web resources in all application domains including AR. Sophisticated massively multiplayer online worlds were now realizable and one example was “Second Life” ("Linden Lab: Makers of Second LIfe," 2011). Rather than a game, Second Life provides a MMO sandbox where thousands of users can build and interact with a virtual world of their creation. The idea of utilizing The SL platform for AR came about when the client source was released, meaning that it was possible for external groups to create new types of interfaces to the server backend. Dr. Michael Nitsche in the School of Literature Communication and Culture was already using SL for machinima and this was the impetus for a collaborative project with the AEL exploring SL for AR.



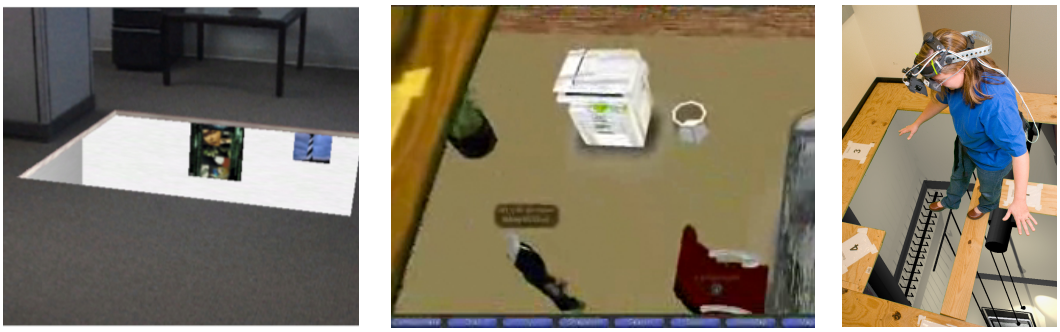
(a) (b)

**Figure 33.** (a) The AR Stages authoring environment inside of the Second Life world (b) A user's view of ARSL. An avatar from Second Life is registered and integrated with the physical scene.

On the College of Computing side, Blair MacIntyre led the project with graduate student, Tobias Lang, filling the role of lead developer. The goal was to achieve the ease of authoring that came with DART coupled with an MMO platform to support multi-user systems and an accessible content creation mechanism, two features that DART lacked. An ARSL client was created and used to quickly explore a variety of content-centric applications. The authoring tool was based around the notion of "AR Stages", persistent, evolving spaces, which encapsulated AR experiences in online 3D virtual worlds (Lang, MacIntyre, & Zugaza, 2008). The design philosophy was that in an MMO, AR experiences are less like "applications" that are designed, loaded and executed, and more like an "installation" that is created, evolved and performed in a defined area in the virtual world. This approach was inspired by all our previous experiences with DART and focused on *accepting the paradigms of an authoring culture* and *attempting to conform to those existing practices*. This goal of exposing AR technologies via conventions already accepted by a design community was evident in ARSL, where SL users/authors were accustomed to simple in-world content creation tools based around primitive virtual objects (i.e. "prims"). The challenge of this project, like those preceding

it, was to expose complex AR topics like tracker transforms, cameras, and physical objects for occlusion via already understood SL metaphors.

In DART this was accomplished via the fundamental Director authoring metaphors of the graphical timeline and stage. We crafted AR components to be exposed via the timeline, despite the fact that in some cases it might have been more efficient or "correct" in a 3D graphics, software development, or AR research sense to handle them differently, but the score mechanism was something Director developers understood. In this same spirit, ARSL embodied AR technology in the 3D scene, since that is where SL authoring took place. The virtual world provided a spatial setting for these AR stages, along with a surrounding area to create the content and behavior for them. Virtual objects in the world represented components such as markers and concepts such as the 3D clipping plane; properties were set by defining parameters on the virtual objects and spatially defining relationships between them (see Figure 33. (a)).



(a) (b) (c)

**Figure 34.** The AR "pit" experience built in three generations of AEL authoring tools (a) a prototype built with DART (b) and alternate version of the experience built with ARSL (c) the full version of the application built with UART

ARSL provided SL appropriate versions of many DART concepts and was informed by our *AR authoring tool guidelines*. For example, one motivation for using SL was that, like Director, developers could leverage and existing, *active developer*

*community*. Like DART, ARSL acknowledged the *importance of the physical world* in AR design; objects in the SL scene could represent either virtual content or physical objects, making it easy for a designer to build an experience that integrated with the physical world in sophisticated ways (see Figure 33. (b)). The custom ARSL client used VRPN to provide *hardware access* to a wide variety of trackers and sensors. *Configuring and debugging* an experience was approachable due to visual widgets designed to fit the SL authoring paradigms, which, also made it relatively easy to *swap between different tracking solutions*. The *content pipeline was approachable*, more so than DART, to novices via the in-world modeling based around "prims", but could be used by experts to create sophisticated objects. The fact that SL was an MMO and ARSL development took place "in world" meant that it was uncomplicated for a large number of team members to *collaborate and even work in parallel*. This access and collaboration was made possible by the *layered authoring* offered by SL, ranging from the ability to build a custom client application, to the ability to place simple primitives in a virtual world with no programming.

This use of the virtual world as the authoring platform also meant that it was possible to interact and observe an application from the standard SL VR client. Multiple times we controlled and *monitored deployments* of ARSL experiences via this approach. It was even possible for stakeholders to observe a deployment from remote locations. As with DART, ARSL was particularly powerful for *rapid prototyping* and we used it to create early explorations of a variety of AR experiences. For example, Figure 34. (a) shows a very early DART prototype of an AR "pit" experience that led to an AR experimental test bed (see Figure 34. (c))(Gandy et al., 2010). Later we were able to use ARSL to very quickly create two alternate versions of the pit, an office environment (see Figure 34. (b)) as well as a version where the user was looking down onto a mountain landscape (Lang, 2007). In both cases we were able to explore dramatically different

versions of the experience with a small outlay of resources and informally test both of these environments with users.

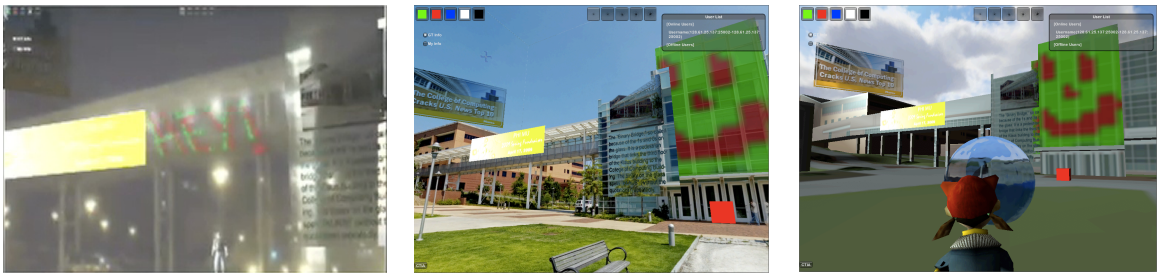

(a) (b)

**Figure 35.** The marker-based ARSL client (a) A user interacting with ARSL objects on the tabletop (b) the augmented view of the scene

Mirror Worlds: the bridge from ARSL to Argon

A main theme of the original ARSL project was about mapping segments of the virtual SL world onto physical analogs. This concept of *connecting the virtual and physical worlds* led into a multi-year research project focused on "mirror world" concepts (Hill, Barba, MacIntyre, Gandy, & Davidson, 2011). A mirror world is an environment where virtual layers of information are directly tied to the physical world and interactions can occur from either side of the reality continuum (Murphy, Kahari, & Ville-Veikko, 2010). This type of research would have been extremely difficult years before, but the proliferation of web-based data sources, the wide use of social networking, and the tremendous increase in geo-coded data meant that it was possible to begin creating true mirror worlds, and powerful mobile devices meant that there were users interested in, and capable of, accessing them. At first, we utilized the ARSL engine as it supported rapid development of mirror world prototypes. A mobile tabletop marker-tracking based implementation (see Figure 35. ) allowed us to examine research questions related to the

representation of MMO objects in the AR scene, tangible interactions, navigation, and the AR presentation of the MMO world at different scales ranging from life size (see Figure 33. (b)) to a World-in-Miniature (see Figure 35. ).



(a) (b) (c)

**Figure 36.** Three views of the Unity3D based MirrorWorld (a) the AR version viewed at night on the Georgia Tech campus. Virtual graffiti drawn by remote users cover the buildings. A virtual avatar controlled by a remote user is visible in the bottom of the screen (b) The panorama-based version (c) The pure VR view of the world. When the avatar is guided into the bubble in the middle of the scene the view switches to panorama mode.

However, over time, the SL system proved too limiting. While the ease of content creation made the system accessible to the novices it was difficult to load sophisticated 3D models. There was also a lack of control over the system, which is entirely defined and controlled by Linden Labs (e.g. forced client updates would make our custom client unable to connect to the network). Also, the server-centric system caused slow rendering performance and latency in interactions. Lastly, links to live data sources, crucial in a mirror world, were not well supported in SL. This led to the next version of the mirror world platform, which we developed in the game engine, Unity3D ("UNITY: Game Development Tool," 2011). We once again created AR appropriate components for an existing engine informed by DART; this Unity system was called UART ("Unity AR Toolkit (UART)," 2011) (see Figure 34. (c) for a version of the AR “pit” built using UART). With UART the goal was to provide some of the developer experience of DART

with more control over the rendering engine. Unity3D was ideal for this, as it was created by a subset of the developers responsible for Director 3D engine. AR technologies (i.e., trackers and cameras) were exposed via Unity's graphical editor. UART did not recreate all the ideation and rapid prototyping features of DART, but was instead focused on *hardware access*, as well as the *modern, mature content pipeline* and *layered authoring* environment provided by Unity3D. UART was utilized to create several mirror world prototypes culminating in a version that used a DarkStar server to provide a persistent MMO world.

Our experience from the ARSL client to the multiple Unity versions highlighted one particular research issue that proved extremely influential on our current work. We began to think more formally about the fact that for mirror worlds to be useful they must provide a variety of user experiences that fall all along the continuum from the *purely virtual to the entirely physical*. These two layers can be combined in a variety of ways, which is determined by the location, device, and needs of the user at the time. No one presentation is ideal, but depends on the tasks the user is performing. As a result, in the mirror world projects we designed a full AR mobile client for use at the physical location, a mobile client utilizing panoramas to support interaction offsite, and a purely virtual desktop client allowing remote visits to the mirror world (see Figure 36. ). This theme has become central to our present AR research.

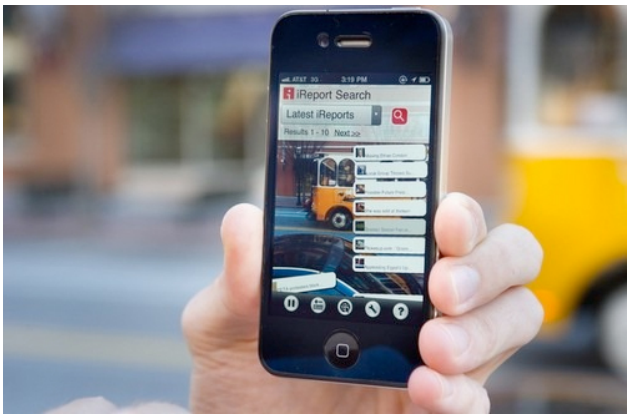
This work also caused us to face multiple server and data issues. The result was that we were ready to confront the challenges of making these types of applications extensible and truly deployable. It was this research goal, which led us to realize the need to fully integrate our AR authoring and infrastructure with the web and the result was the AR web browser, Argon.

## Argon: An AR Web Browser

The impetus for Argon was that we, and others, realized that *web technology* was becoming capable enough to serve as a platform for AR and mobile devices. While even the earliest AR systems such as The Touring Machine (Feiner, et al., 1997) required large data sources that ideally would have resided "in the cloud", the systems envisioned custom server solutions as the web either did not exist in its current form or, in later years, did not provide the location aware, API accessible, and ubiquitous availability that was required to make such applications a reality. Our move toward the web was influenced by the ARSL and Mirror World work and was informed by our early exploration of "big ideas" of AR, such as those discussed in Chapter 3. We had experimented with using Unity3D to look at the intersection of AR and the Internet; we were ready to confront the access to "big data" that was required to build real versions of the AR applications the research community had always envisioned. Once again we were designing an AR authoring environment, but in this case it was to be informed by the *workflow of web developers*. From our work with DART we realized the value of focusing on how people that are designing and developing with our tools will use them to really do things. This research philosophy has culminated with the ongoing design and development of Argon.

DART was one of the first AR authoring tools that allowed non-technologists to create sophisticated, useful, and deployable applications like those described in Chapter 6. Since DART was created, web scripting, browsers, content pipelines, servers etc. have become ubiquitous and accessible. People from all professions and from all levels of training in programming and graphics now design web sites. This provides the ultimate ecosystem on which to base a modern AR authoring and deployment system. We chose to base Argon on the basic web building blocks of HTML, KML, and JavaScript (MacIntyre, Hill, Rouzati, Gandy, & Davidson, 2011). Argon "channels," which encapsulate the AR application, are the equivalent of a website and are accessed through

a URL. Argon itself is a native application that is constructed out of low-level access to device functionality (e.g. live video, GPS, and compass) and mobile web views.




(a)

(b)

**Figure 37.** Argon applications (a) an outdoor information browsing application accessing CNN iReports data (b) an architectural visualization on a construction site. The application is authored using a geospot. The virtual imagery is overlaid on a panoramic image rather than live video.

As DART was designed to fit within the interactive media designers' process, and the AR SL system was based on the AR Stage in keeping with the SL approach to authoring, Argon is designed to integrate with the web developer workflow. The use of web technologies has provides features reminiscent of those from DART, including high level script *access to AR technologies* such as tracking, a *vast developer community*, *layered authoring* that leverages existing web development tools and languages (e.g. JavaScript, Google Earth, Dreamweaver), a *sophisticated and approachable content pipeline*, *ease of application iteration,* and *content substitution for prototyping*. This also allows Argon developers to leverage the vast amount of APIs for interconnecting systems and to access live data sources, thus supporting large-scale deployment (see Figure 37. ).

A core feature of Argon is allowing us to continue our research into the virtual/physical continuum and the variety of AR/MR/VR presentation modes that allow users to consume the mirror world content in multiple ways. This feature, called a "geospot," is in the tradition of our previous work. It allows authors to define known locations in the world from which a user can view content. One or more panoramic images can be associated with a geospot and are used in place of live video. This feature evolved from an acknowledgment of the current constraints of AR tracking technology. However, over time we have realized that the geospot also makes the process of placing 3D content in the world more approachable, as the author can define 3D object transforms relative to geospots. We also have observed current students using this feature in ways reminiscent of the "video prototypes" students of the past created with DART (i.e., the application is spatially "incorrect" but produces an experience that looks correct with the captured content). This relatively low-tech addition to the environment supports *early ideation and rapid prototyping* as well as *debugging away from the physical site*, related to how we previously utilized capture/playback features of DART in locations such as Oakland Cemetery.

Our *collaborations with designers* continue with Argon. Currently, a group of LCC students, led by Jay Bolter, are creating a variety of tours and art installations utilizing Argon and geospots. This work includes revisiting our previous "Voices of Oakland" project (see Chapter 5.) that was originally prototyped and evaluated using DART ("Voices of Oakland," 2011). Now with Argon this experience can be deployed to real users.

Argon developers are starting to explore the creation of *meta-tools*, simple high level authoring environments tailored for a narrow domain, a need discussed in Chapter 6. The web development model and high level tools make it possible to author such tools quickly. For example, the ARBox project, built by a student team in a semester, provides

a World Board-style (Spohrer, 1999) system for making simple AR tours based on placement of virtual content cubes without programming (Su & Feng, 2012).

Argon does not yet meet all of my AR guidelines. For example, tracking is currently limited to GPS + orientation and vision-based markers; there is not yet a unification of coordinate systems that supports painless swapping between technologies. However, *hardware access* via VRPN is soon to be added. Argon does not yet contain features to explicitly support *ideation, rapid prototyping, or evaluation*. Nor are there high level features designed to assist in *debugging*. There is also little concept of the physical world in Argon, beyond geo-locations; the existing applications emphasize placing virtual html divs on top of the scene rather than *tightly integrating the virtual and physical*. However, the goal is to incorporate more of these features over time. The process of using DART and the feedback from external developers revealed some authoring requirements that DART did not adequately address. Argon is an opportunity to explore features that *convey affordances and constraints*, *support smooth transition of early ideation artifacts into technology prototypes*, and *assist with identifying the cause of application errors*.

We have come full circle, back to many of the application ideas that we explored in the early application-building era of our research. I am interested in exploring sound interfaces for Argon, informed by early audio AR project such as GBV (See Ch. 3). We are currently adding features to Argon that expose this high level control of audio presentation to the author. With the advent of Argon, now many of these AR application ideas that we and our AR research colleagues have prototyped or imagined for years can be developed and deployed, leveraging all the content and services of the web. Argon allows us go to next level of AR research, exploring what happens when we put these "big ideas", driven by live data, in the hands of thousands of real users.

# CHAPTER 8

# CONCLUSION

My thesis was focused on investigating designer workflows and goals to inform AR authoring tool guidelines and an authoring tool based on this research that allows creators with a range of technical expertise to explore the AR medium. In this dissertation I have traced a decade long research thread focused on developers with diverse skills and objectives; demonstrating how this research allowed us to create tools that effectively advanced AR.

There are four main contributions presented in this thesis:

- Exploration of established work flows and AR authoring needs via collaborative research and development projects
- The development of an authoring tool for non-technologists (DART) informed by these collaborations
    - The identification and validation of AR authoring tool guidelines which resulted from the process of creating and using DART
- A study of long term DART use by external developers. This work helped to validate our guidelines and the value of DART, while also contributing new information on the needs for future tools
- Significant impact on the current and future research of the AEL. The influence of the guidelines and DART on future research is a validation of both, just as the external study and internal application building was. This impact further validates DART by showing how the results have carried forward to a new generation of research.

In the early AR development projects I worked with diverse teams on collaborative AR projects that were exploring the use of AR for entertainment, performance, and narrative experiences. These projects explored the "big ideas" of AR while conforming to the limiting technology restrictions of the day. These projects resulted in more than software and hardware artifacts. They helped me to identify some initial requirements for AR authoring tools. These projects highlighted the importance of *approachable hardware access*, the need for tools that allow *diverse teams to collaborate*, and the demand for *mature content pipelines* that leverage existing tools and workflows. These projects also revealed the *significance of rapid prototyping* and how essential it is to *evaluate the entire the user experience early and often*. Lastly, these experiences showed to me the value of *leveraging existing media development tools* for AR authoring. Overall, this work directly informed the next contribution, which was the development of DART.

The Designer's Augmented Reality Toolkit encouraged developers from outside the AR research world to engage directly with the medium.

Our goals for DART were to:

- Support the entire design process
- Provide a powerful easy to use development environment
- Ameliorate the problems of working in the physical world.

We designed novel AR specific authoring features such as the capture/playback architecture, WoZ support, and sketch based prototyping to address all four stages of the process (i.e., idea exploration, populating the virtual world, application development, and deployment & evaluation).

A research contribution from this work was a set of AR authoring tool guidelines:

- Support *rapid prototyping* of the virtual and physical worlds
- Help designers *transform early ideation artifacts* into initial technology prototypes
- Encourage *participation by various project stakeholders*
    - Provide a *layered authoring environment* that lets contributors with different goals and expertise engage with the authoring process
- Establish a *mature content pipeline*
    - Allow for the use of *proxy content* early in the process
- Permit developers to *access arbitrary hardware* for tracking/sensing
    - Make it easy to *switch between technologies*
- Assist with the *debugging process*, including the task of identifying the source of errors before crafting a solution
- Support *monitoring* and debugging of live deployed systems
- Encourage *evaluation* via mechanisms for supporting studies, logging, and visualizing application data.

DART was used to build a wide variety of internal AR and MR projects which exercised its full range of capabilities. These experiences revealed the value of DART features including layered authoring, the content pipeline, capture/playback, WoZ support, and the tracking/sensor architecture. They illustrated the value and challenges of collaborations with diverse teams and the overall power and flexibility of DART. Our experiences with these projects demonstrate the validity of our authoring tool goals, development stages, and guidelines presented in Chapter 4.

Over a period of six years DART was downloaded by thousands of users and was utilized for a large number of projects. I gathered reflections from a collection of our external developers that used DART to build sophisticated and ambitious AR systems.

These developers represented the full spectrum of expertise, from computer science graduate students to a theater director, as well as a range of project goals, including formal AR experiments, AR education for humanities students, and art installations. Their experience with DART validated my AR authoring tool guidelines and introduced new concepts including the need to *convey the affordances and constraints* of AR via the tool and documentation, the challenge of helping those inexperienced with computational thinking to *realize the power of sophisticated prototyping and debugging features*, the importance of a *developer community*, the need for tools to provide easy mechanisms for *reusing and repurposing portions of projects*, the challenges of *debugging system components* outside of the AR authoring software, and the desire for *meta-tools* that would allow developers to quickly build simple project specific authoring environments.

From the internal and external projects I learned that DART was successful at allowing people (even those who were not "computational thinkers") to quickly develop robust and complicated AR applications. The support for rapid prototyping was unique to DART, "*raising the ceiling and lowering the floor*" of AR for a large number of creators, and *bringing HCI techniques from other research domains into AR authoring*. The layered access allowed diverse teams to work together effectively. Overall the feedback regarding the utility of DART was overwhelmingly positive.

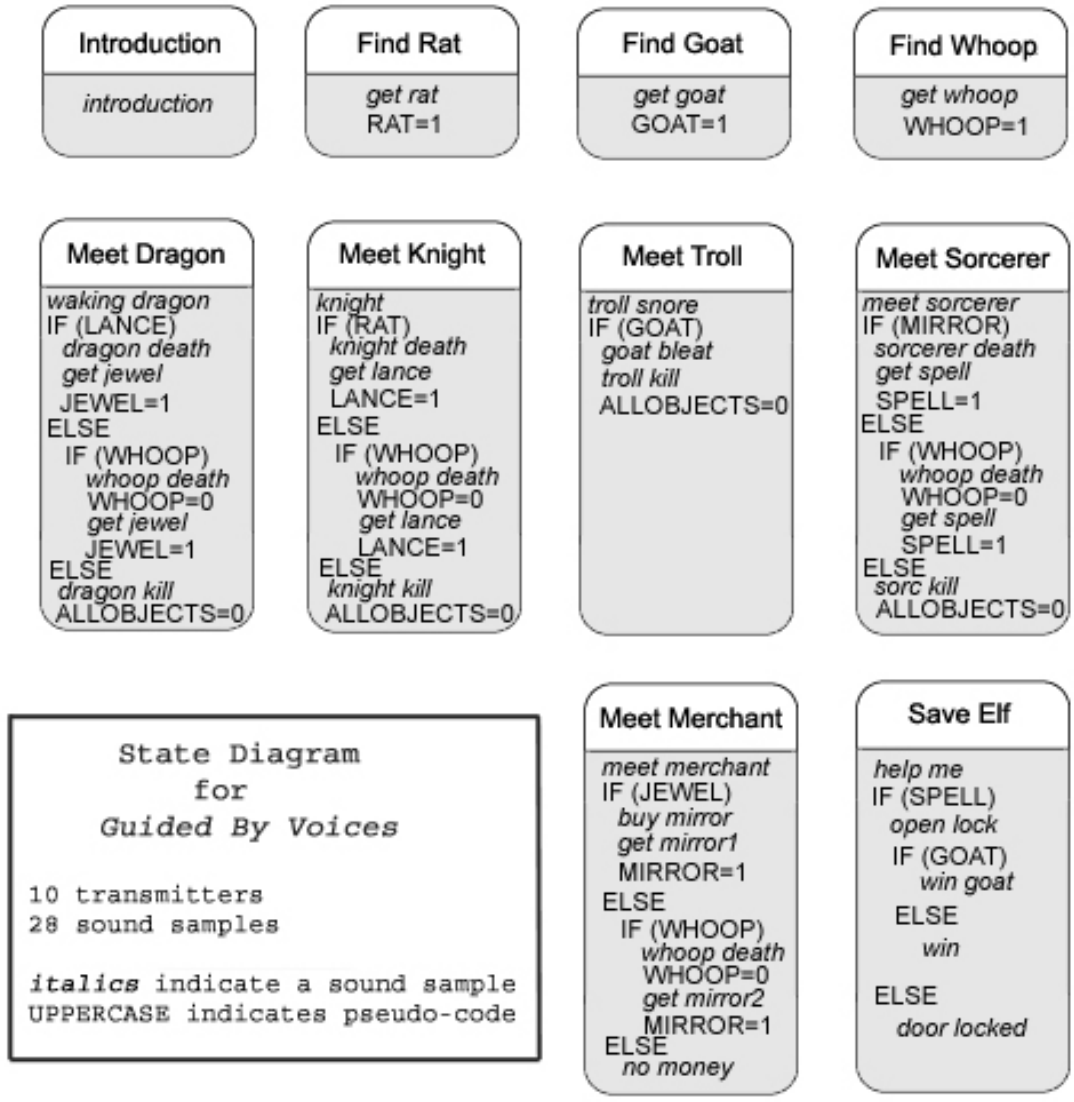
The impact of my work with AR authoring, informed by designer workflows and needs, can be traced through to present day tools at Georgia Tech. We have continued to be informed by the guidelines for AR authoring and the project goals that first defined DART. The ARSL project appropriated and molded them to fit a different type of environment, the Second Life MMO, and the workflow associated with it. While the experience of authoring in ARSL was quite different than DART, it was designed around the same principles of providing *rapid prototyping support*, *approachable access to AR technologies*, and a *mature content pipeline*. This work led to the development of DART inspired elements within the Unity3D game engine-based UART. Now a modern

approach, developing AR authoring tools based around web development technologies, is being explored via Argon.

My addition of AR scaffolding in real development environments such as Director, Unity3D, and, now, the World Wide Web enabled a range of contributions to the field of AR research at Georgia Tech and elsewhere. This approach is proving increasingly relevant, as AR becomes a mainstream medium. There is an ever more diverse population of people working in the AR space now, including entrepreneurs building companies based on the delivery of AR content ("Layar," ; "Merlin Mobility - Augmented Reality Instructions," 2011), artists leveraging AR in their work (Oliver, 2008), technologists developing sophisticated solutions for tracking and displays ("Qualcomm Augmented Reality SDK," 2011; "Vuzix - View the Future Today," 2011) and game developers (McFerran, 2011). Ambitious projects are coming from industry that, like Argon, aim to unify location-based and MR applications with the web ecosystem via standardization of formats and protocols. Microsoft's Read/Write World initiative aims to index, unify, and connect the world's geo-linked media; providing access via open source viewers and real-time geo-services ("Read/Write World | A flexible fabric for exposing, connecting, and consuming geo-media and geo-data," 2011). This change of scale for AR applications requires more complicated workflows, which we will continue to address via our research approach of creating tools informed by the needs of practitioners and designers.

# APPENDIX A

# STATE MACHINE OF GUIDED BY VOICES ROLE-PLAYING GAME


Introduction
introduction
Find Rat
get rat
RAT=1
Find Goat
get goat
GOAT=1
Find Whoop
get whoop
WHOOP=1
Meet Dragon
waking dragon
IF (LANCE)
dragon death
get jewel
JEWEL=1
ELSE
IF (WHOOP)
whoop death
WHOOP=0
get jewel
JEWEL=1
ELSE
dragon kill
ALLOBJECTS=0
Meet Knight
knight
IF (RAT)
knight death
get lance
LANCE=1
ELSE
IF (WHOOP)
whoop death
WHOOP=0
get lance
LANCE=1
ELSE
knight kill
ALLOBJECTS=0
Meet Troll
troll snore
IF (GOAT)
goat bleat
troll kill
ALLOBJECTS=0
Meet Sorcerer
meet sorcerer
IF (MIRROR)
sorcerer death
get spell
SPELL=1
ELSE
IF (WHOOP)
whoop death
WHOOP=0
get spell
SPELL=1
ELSE
sorc kill
ALLOBJECTS=0
State Diagram
for
Guided By Voices
10 transmitters
28 sound samples
italics indicate a sound sample
UPPERCASE indicates pseudo-code
Meet Merchant
meet merchant
IF (JEWEL)
buy mirror
get mirror1
MIRROR=1
ELSE
IF (WHOOP)
whoop death
WHOOP=0
get mirror2
MIRROR=1
ELSE
no money
Save Elf
help me
IF (SPELL)
open lock
IF (GOAT)
win goat
ELSE
win
ELSE
door locked

# APPENDIX B

# AUGMENTED REALITY DEVELOPER QUESTIONNAIRE

1) Had you built AR applications before your use of DART? If not, why?
    a) What types of applications did you build?
    b) What tools did you use?
    c) Did those tools meet your needs?
2) Did you have any previous experience with Director? At what level?
3) What led you to choose DART for your AR authoring?
4) Describe how you used DART
    a) when and for what time period?
    b) What were you overall thoughts about the experience?
5) What were the biggest challenges you encountered during AR authoring?
6) Describe how you used DART during the four parts of authoring process
    a) Exploring Ideas (e.g. brainstorming, early prototyping)
    b) Populating Virtual World (e.g. prototyping, user experience testing)
    c) Developing the Application
    d) Evaluation and Deployment
7) What problems did you encounter working in the physical world (brainstorming, debugging, tracking, deployment etc.)?
    a) How did DART assist or not with these problems?
8) What features of DART did you use? (e.g. Capture/Playback, SketchActors, VideoActors, switching between trackers during the process, Wizard of Oz)
9) Were there features of Director alone that you felt were particularly powerful in your authoring process?
10) What other tools did you use in the authoring process (e.g. Maya, Photoshop etc.)? How were they utilized? Did you encounter problems utilizing content from these tools in DART/Director?
11) Were there features of Director alone that you felt impeded your authoring process?
12) Did your finished AR experience meet your goals? In what ways did it fall short?
13) Did you extend DART or Director? Describe.
    a) edit built-in scripts?
    b) write new scripts?
    c) edit/build Xtras
14) Were there features of DART that you felt were particularly useful in your authoring process?
15) Were there features of DART that you felt impeded your authoring process?
16) What changes or new features of DART would have improved your authoring experience?
17) Have you built AR applications since your use of DART? Which ones?
18) Have your AR authoring needs changed/evolved since you used DART? How?
19) What authoring tools are you currently using for AR development?
20) How do these tools differ from DART?
21) Are there authoring needs not being met by these tools?
22) Describe what features your ideal AR authoring tool (for your current projects) would have.

# APPENDIX C

# DART OVERVIEW DOCUMENT 3.0

The Designer's Augmented Reality Toolkit (DART) consists of a set of Lingo scripts and an Xtra plug-in that extends Macromedia Director to support the development of Augmented, Virtual, and Mixed Reality applications. We chose to develop on top of Director as it provides a very full featured development environment with an active developer community and cross platform support (Win and Mac/OSX), something that academic software projects do not often have. By leveraging the features that already existed in Director, including a powerful 3D engine, we have focused our efforts on integrating the necessary AR components with Director's programming model. The Director environment provides programmers with pre-built scripting components (in the Lingo scripting language) that can be used as is, or easily modified and extended by the developer. All the scripts are open and editable, allowing a developer to easily create new components as needed. This type of extensibility is crucial in a toolkit such as DART since it is impossible to anticipate and provide support for all the applications the users will want to develop. Thus, while interesting applications can be created in DART with little or no programming, you should really view DART as a starting point for application development, and the included scripts and behaviors as a model for your own development.

The Xtra is a plugin for Director written in C++. This plugin adds low level AR related functionality into Director including video capture (from a variety of cameras, currently including all Quicktime cameras on MacOSX, and on Windows those supported by DirectShow plus custom libraries for the Videre Design DCAMs, PointGrey firewire cameras, and soon Canon DSLRs), connection to VRPN sources (trackers, buttons, analogs and distributed shared memory objects), fast video-mixed AR via OpenGL, and marker tracking (ARToolkit and ARTag).

The behavior scripts (written in Lingo, the Director programming language) are part of the Director authoring environment and can be manipulated just like Director's built-in components. The DART behaviors represent the high level components that make up an AR application, and provide structured access to the various AR technologies. There are *Actors* which represent the content of an application (3D models, meshes, particle systems, 2D heads-up-displays (HUDs), sounds (ambient and spatialized), and lights) and a *3DCamera* that represents the virtual camera in the 3D world. *Textures* allow the designer to texture video, sketches, flash movies, text, and live video on the Actors and *Shaders* let the designer use different materials and rendering modes for the 3D Actors. There are behaviors that connect into the functionality of the Xtra such as *LiveVideo* which configures the camera to be captured, and *LiveTracker* which represents a tracker (VRPN or marker) in the application. *PlaybackTracker*s and *PlaybackVideo* behaviors allow you to use prerecorded video and tracker data in the application as though it is live. *Transforms* are placed on *Actors* and *3DCameras* to control their position, orientation, scale, and to define the parent/child relationships of the scenegraph. *Transforms* can subscribe to *Trackers* (both Live and Playback) which is how the connection is made between objects in the application and trackers. The scripts have been designed so that AR applications can integrate whatever technologies they need, easily mixing 3D and 6D trackers with marker tracking and other sensors.

Interactivity in DART applications is achieved via a cue/action model. *Cues* are events that are fired when things happen in the application (e.g. the *3DCamera* reaches a certain position, an audio clip finishes playing, a timer reaches a defined value, a marker appears or disappears, etc.) *Actions* subscribe to cues and wait for them to occur. When the specified cue fires, the action will execute (e.g. start an animation on an object, move an Actor in the 3D world, change the volume on an audio clip, etc.)

The development of a DART application starts just as any Director application would, with placing components on the Director score. Typically Director components are sprites which can be placed on the score and configured via a property page. Unfortunately the 3D world does not provide individual sprites to control the scene. Instead the 3D world is a single Director media element, a black box which requires Lingo programming to control. Therefore, we do not have 3D sprites to use as the basis of the DART components, nor have we created custom sprites. Instead, we use standard text sprites as "containers" on which you can place the DART scripts (encapsulating the 3D functionality as well as the other components such as Actors, cues/actions, trackers etc.) These container sprites hide behind the 3D world on the stage, but allow DART developers to leverage the facilities of Directors score for content organization and application control flow.

The DART scripts themselves are contained in a series of "casts" which must be loaded in a DART Director application. These casts divide up the scripts into logical groups such as Actors and Events. One cast ("*DART-Core*") contains movie scripts which should not be placed on the score and do not have properties to set, but must be present in all DART applications. These are global scripts that provide the core runtime of DART, handling various behind-the-scenes processing for DART applications such as the tracker and sensor subscription management and the cues/action broadcast system.

To develop a DART application you place scripts either on the 3D world sprite (provided with Director) or you place them in the container sprites. When you place a script on the score it brings up a property page where you can configure the behavior. In this manner you can build up the parts of an AR application. For example, assume you wanted to create an application with a sphere sitting in the middle of the world. First you place a *3Dcamera* script on the 3Dworld (a Shockwave3D sprite) along with a *Transform* script to place the camera at an x,y,z of (0,0,200) in the scene (out of the z-axis, looking at the center of the world). Then you place an actor container sprite on the score. On that sprite you place an *ObjectActor* behavior and in the property page indicate that this object should be a sphere of size 20. You also place a *Transform* on the actor sprite and configure it to place the *ObjectActor* at an x,y,z of (0,0,0). You will also need to place a *DART-Loop* script on a frame of the score above where your sprites are placed. Since AR applications (and in fact most interactive applications built in Director) are not linear you do not want the play head to simply move along the score. Instead you lay out logical sections of your application on the score and have the play head loop on a single frame of each section. Jumping between sections can be accomplished via cues and actions. This *DART-Loop* script on the frame will simply cause the play head to stay on this one frame and functions as the main loop of the application. When you run this application you will see a sphere sitting in the middle of the stage. Although this is a simplistic example, all DART applications can be built in this manner; placing containers sprites on the score and filling them with the desired behavior scripts. For more advanced applications the developer can simply edit code in the behavior scripts, add new scripts of her own, or copy an existing script to serve as a starting point for a new component. DART scripts and custom Lingo can be freely mixed together.

This overview has only touched on the most basic portions of DART. Other advanced features include
- DART now fully supports all the features of Director 3D textures, so the various types of textures (video, flash etc.) can be used in sophisticated ways.
- Support for Wizard-of-Oz control of applications (a person can fire cues and control a DART application manually from another machine).
- A Capture/Playback system that allows a developer to capture live video, tracker, and other VRPN data and then replay it in their applications as if it were live (enabling video storyboarding, development, and debugging away from the work site).
- The ReplayAR script that provides Tivo-esque control over live AR applications.
- A collection of Physics components that are used to give the Actors in an application physical properties (mass, friction, elasticity) that are then controlled via the Havok physics engine provided with Director.
- A cast of debugging related scripts that can be used to control the master clock, move the virtual camera, observe tracker data, and to configure the marker tracking.

To begin exploring all the features of DART try out the various example applications or "template apps" that are included with the DART distribution. There is a template app for every DART script.

For a more detailed look at the research behind DART, please read our UIST paper, available at http://www.cc.gatech.edu/ael/papers/dart-uist04.html:
Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. "*DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences*." Proceedings on conference on User Interface Software and Technology (UIST'04), October 24-27, 2004, Sante Fe, New Mexico

# APPENDIX D

# DART HIGHLIGHTS 3.0

- In addition to bug fixes…
- The biggest change in this release is how most of the former "actor" functionality is now abstracted into various "texture" scripts. This means that videos, sketches, flash movies etc. (which used to be encapsulated by "actors") are represented by textures that can be placed on 3D objects (ObjectActor), 2D heads up displays (HUDActor), or on particle systems (ParticleActor). These textures are available in the new DART-Textures cast. These texture objects give you much more control over how textures are placed on objects and opens up the full functionality of Director3D textures to the DART user. Textures can also be applied to models or shaders on an individual basis allowing for sophisticated use of the texture objects even with 3D models. Please explore the template apps for DART-Textures to see the full range of options. The new textures scripts include:
    - Texture: the basic texture that lets you texture still images on an actor
    - VideoTexture: textures video on an actor
    - SketchTexture: textures animated images on an actor
    - TextTexture: textures text from text cast members on an actor. FlashTexture: texture interactive and non-interactive flash movies on objects
    - LiveVideoTexture: texture streams from a live camera on an object. Changes have been made in the LiveVideo and PlaybackVideo scripts to allow the streams to be named and distributed to texture subscribers.
    - A TextureActions script allows the user to change texture settings on the fly in the response to cues. These actions include the ability to enable textures, modify text properties, change the images used in a texture.
    - When setting values for textures the user can define them explicitly in the texture properties or can allow the current default values in the shader to be used.
- As a result of the new texture paradigm many of the old actors have been deprecated (e.g. VideoActor, SketchActor, FlashActor etc.). You will notice that these scripts are still in the release (their names are now preceded by "old_") in order to preserve backwards compatibility with older DART applications. However, in the future please avoid using them as they will be removed in the next release. Three new actors have been created that can operate with the texture scripts: ObjectActor (as before, used for 3D models and primitives), HUDActor (creates 2D heads up displays), and a ParticleActor (creates particle systems).
- There is a Shader script available in the DART-Textures cast that lets you create new shaders and place them on ObjectActors. The user can also control texturing parameters from the shader object instead of from individual textures if desired. There is a ShaderAction script that allows you to change some shader properties on the fly in the response to cues. The support for more actions can be added to this script quite easily.
- The ObjectActor can now be used to create meshes. The user defines the vertices and faces in an internal text file. This definition is loaded by the ObjectActor and used to create the mesh.
- The old version of the AudioActor (now called "old_AudioActor") has been replaced with a new version that handles both ambient and 3D sound sources. One audio actor can be switched between ambient and 3D by simply clicking a checkbox in the property page. When a sound is spatialized it can be controlled by a Transform script just like any other 3D object in DART. Please note however, that only mono wav files can be spatialized, quicktime files can only be used as ambient sources. Also, the 3D sound will only work on Win platforms as the OpenAL xtra is currently only available for Win and not MacOS.
    - Video and SketchTextures play synched audio along with their visuals and therefore these sound sources can be spatialized by simply changing a checkbox in the Video and SketchTexture property page. Remember though that you must use wav files in order for the spatialization to work
    - A 3DAudioProperties script can be placed on an AudioActor, SketchTexture, or VideoTexture to control more properties of the 3D sound such as sound cone and volume.
    - A 3DAudioInit script is required when 3D sound will be used. It sets up the global OpenAL parameters.
- Better synch of activities on prepare, enter, and exit frames.

- HUD depth. User can now specify a distance from front or back clipping plane for HUDActor, LiveVideo, and PlaybackVideo overlays.
- Some changes have been made in the LiveVideo script and xtra that will provide multi-camera support in the next release.
- Independent control of FOV settings. The user can specify different FOVs for the 3DCamera and the background video. The user can also have both of those FOVs set to that of the live camera (this is defined in the LiveVideo property page).
- Video configuration settings for a DART movie can now be specified via an external text file called "camera_config.cfg." The LiveVideo simply looks for that file in the current working directory; if it finds the file the settings specified in that file supercede those in the property page. This allows you to use your application with different cameras easily. For more information on this please see the VideoStub template app in the Template-Apps/Projector directory.
- The VideoStub template app in the Template-Apps/Projector directory lets you run .dir movies as standalone executables without having to "publish" it. The videostub also lets the user specify camera settings at run time. For more information on this please see the VideoStub template app in the Template-Apps/Projector directory.

Known Problems

- On the Mac if the Director file is anywhere under your home directory, automatically setting the path to the Data directory does not happen correctly if the path is left blank in the LiveVideo script (not clear how to fix this; contact us if you would like to try and figure this out).
- 3D audio will not work under Mac OSX (the Xtra has not yet been ported)
- Camera resolutions greater than 512 x 512 are not supported on Mac OSX (seems to be a limitation of texture sizes in Director on the Mac) (this will be fixed shortly, by using multiple textures and a polygon mesh).
- ReplayAR does not yet capture buttons, analogs, or shared memory.
- After running a movie that uses the Xtra sometimes drag and drop will stop working in the Director authoring environment (restarting will fix this, seems to happen primarily when using Directshow cameras on Windows).

# APPENDIX E

# DART TIPS 3.0

Want to become a DART guru? Want to debug your DART applications at the speed of light? Check out this list of tips that we have gathered helping others use Director and DART.

1. **If you are unfamiliar with Director, do the standard tutorials before you use DART.** There are some simple tutorials for Director in general and the 3D functionality in particular that come with the program as well as many available on the web (http://kingkong.cc.gatech.edu:8080/ARDesign/238 is a Director/DART tutorial exercise that we have students do in our AR Design class). Do some of the tutorials to get comfortable with how Director applications are put together. You don't need to be a Director expert to use DART, but you do need to understand the basics of the score, casts, scripts etc.
2. **My camera is not working!** This is the single most common problem people have with DART. This is because the developer needs to set the values in the LiveVideo property page to a format that is supported by the particular camera they are using (e.g. YUV, 320x240, 10 FPS). There are many cameras in the world, especially webcams and they all seem to support a different subset of formats. One way to determine what formats work with your camera is to run whatever video capture program might have come with your camera. Usually there will be a configuration menu somewhere that shows you what format(s) the camera is using. If you are using a DirectShow compatible camera (most consumer webcams) DART will provide you with a list of supported modes. Open up the LiveVideo template app that comes in the DART installation. Run it. If the video doesn't appear on the stage open up the message window (press Ctrl-M). There should be a list of supported formats for you camera. Pick one of these formats and edit the LiveVideo script property page to contain these values. If you continue to have problems send an email to the DART Users mailing list, we are happy to help you resolve your camera issues.
3. **Install the 3DPI xtra.** If you are doing anything with the 3D world in Director 3DPI is a great tool. This is an xtra that you can use during runtime to inspect and modify the 3D world. It makes it so much easier to see what is happening in the scene. http://www.3dpi-director.com/

4. **Recompile All Scripts**. Having some strange error or behavior in your applications? Try running "Recompile All Scripts" from the menu.
5. **Drag and Drop stopped working!!!** Yes, this is a known but mysterious bug caused by the DART Xtra. Sometimes after you run live video capture, drag and drop will stop working inside the authoring environment. You can probably continue on without out it, or just quit and restart Director to fix the problem.
6. **I ran my program but I don't see any of my 3D objects!** This is a common error in a DART application, and there are many reasons why you aren't seeing anything.
    a. Check your units of measurement. DART uses centimeters as its base unit (we did this because the near clipping plane in a Shockwave3D world cannot be any closer than "1" unit, so using feet or meters was impossible).
    b. Make sure your objects have 100% opacity in the property page. Maybe they are there, but are totally transparent
    c. Make sure that the actors are either set to start automatically, or you have set up a cue and action to start them. Maybe they are there, but weren't started and thus are hidden.
    d. Make sure the 3D camera is pointed in the right direction. Depending on your tracker and such, your camera may need to have a local rotation applied. It is very common that when we don't see what we expect it's because the camera needed to be rotated 180 around y. Try adding a bunch of little objects to the world, and see which ones are visible!
    e. Make sure that tracker data is coming in. If your camera is supposed to be tracked, for example, maybe you aren't seeing anything because it is not getting any tracker reports. Or perhaps the object that is supposed to be linked to a marker was inadvertently linked to the wrong panel.
    f. If you are loading a 3D model, are you sure the model is not incredibly huge? Due to the units in DART (as mentioned above, we use cm), sometimes models are very large when loaded. Try setting the scale to 0.1 or 0.01 and try again.
    g. Use the OverlayMapActor. This component will show you a 2D overhead view of the scene. This will help you figure out if objects are actually there and where the camera is looking.
    h. Use 3DPI to inspect the scene (as suggested above).

7. **Use the Message Window.** Most feedback from DART, both error messages (the camera parameters were not correct), warnings (You don't have the OpenGL renderer chosen), and information (a cue has fired, an action has triggered) is printed out in the Director message window. To see the window simply press ctrl-M (cmd-M on the Mac). Whenever something seems to be going wrong, look in the message window first, hopefully DART will provide you with some useful feedback.
8. **Director refuses to use OpenGL, even when I tell it to.** Some graphics cards are very finicky about when they will or will not use OpenGL with Director. The most common (and annoying) cause is that some multi-head cards (e.g., Nvidia cards) will not let Director use OpenGL if you have "multiple displays", but if you set the driver to "span" both displays and treat them as one large display, things will work. This is only a problem on Windows (MacOS always uses OpenGL), and is caused by the way the Windows OpenGL drivers work.
9. **Where is my camera, where are my actors?** If you have a camera or actor that is attached to a tracker, their transformation information will be printed out every 100 tracker reports. This is an easy way to see if tracker reports are being received, and where the objects are being positioned in world coordinates. If you want it to print out more frequently simply open the transform script, look for the line that says "if (whenprint = 100)" and change "100" to whatever number you desire.
10. **What is going on?** Remember, all Lingo scripts are editable by you, so poke around, add "put" statements to see what is being run when. Put in breakpoints and use the debugger.
11. **Use capture/playback for debugging.** When you are trying to figure out a bug, it is very easy to simply capture some data with the condition that seems to cause problems and then use playback instead of live data. This enables you to try lots of fixes rapidly and cuts out wasted time trying to recreate the bug. Most importantly, you can step through the data at whatever speed you want, allowing you to debug a sequence of video frames or tracker reports without missing any. The new ReplayAR script makes it easy to capture data without planning ahead, so you have no excuse not to!
12. **User capture/playback to create simple animations.** Want to have an object hop like a bunny or swim around the room? A simple way to accomplish this is by simply capturing some tracker data where you

move the tracker in the desired animation path. Then attach your object in the live application to the PlaybackTracker using this data. Now your object will move around with your recorded movements. This data can be played back as relative or absolute, so it can control either the world transform of the object, or just the local.

13. **The order of scripts matters.** Director is single threaded. When the play head reaches a set of sprites on the score, the scripts in those sprites are initialized in the order that they are in the sprite and that they are on the score. (The Director documentation explains the order of execution.) Therefore, the order that DART scripts are placed in a sprite is very important. For example, if you have a transform script in a sprite before the ObjectActor script you will get an error; the "actor" script must be the first script on any contain sprite. This is because the transform script will be initialized, and it will go looking for the actor object to which it is attached, however, that object won't have been created yet. We try to account for this linearity (for example, you can subscribe to LiveTrackers that haven't been created yet), however, some problems are unavoidable. The best way to handle this is to look at template apps for the scripts you plan to use, to make sure your order is correct. If you have an unexpected error it is worth looking at the script order and maybe trying a different arrangement.
14. **Can I make a DART application into a standalone executable?** Yes. And it is not very hard. You will need to make a "projector" in Director, such that the entire collection of DART scripts and all the Xtras your project needs are in the projector directory. The project becomes an executable that you can then distribute. For more information on this, please see the project recipe document that we provide on the DART swiki. A simple solution is to use the VideoStub example that is included in the template apps in the Projector directory. You can use this provided exe to run your .dir movie like a standalone program. Please read the readme.txt included in the VideoStub directory for more information.
15. **How do I make my application run in fullscreen mode?** While running your application inside of Director you can switch to full screen mode by choosing "Full Screen" under the "View" menu option. However, if you are using Director MX 2004 you will find that even though it goes into full screen the graphics are always offset from the right hand corner of the monitor. Therefore to have the graphics fill the screen correctly you will need to make a standalone executable (See #14 for information on doing this). Also, remember to

set your screen resolution to the size of your Director stage in order for the graphics to completely fill the screen.

16. **My keyboard presses don't seem to be registering!** For a key press event to fire in Director the stage window has to have the focus. Therefore, try clicking on the stage window to highlight it. Make sure the message window is open since it will show you when key presses are recognized.
17. **I want my objects to disappear when the markers do!** This is simple to do (basically make your application work the way a standard ARToolkit app does). Put a TrackerCue on the LiveTracker script for the marker. Set it to send a cue when the marker is seen and when it disappears. Set up two actions on the actor, one that "hides" the object when the "marker disappears" cue is sent, and one that "shows" the actor when the "marker is seen" cue is sent. Look at the template app for TrackerCue to see how this works.
18. **When I run my application everything is running very slowly!** This is usually due to trackers not working properly deep within the VRPN subsystem in the Xtra. VRPN is the tracker software used by DART, and most of the VRPN server objects that talk to trackers will block while trying to initialize the communication with the object. If you have VRPNInit set to turn on local VRPN servers but you don't have the trackers listed in the "vrpnfile" text cast member connected, or if for some reason the server is not able to find the tracker, there might be a huge slowdown as the server continually tries to reset the device. Try just turning off VRPN to see if that makes a difference, and if it does, start debugging your tracker setup.
19. **When I load my old DART applications with the new release I get these errors about casts moving, and now all the wrong scripts are on the score!** Unfortunately, this is an unavoidable problem with Director. If you build an application with certain scripts, and then you move the position of those scripts inside of their cast, or move them to a different cast, the linkage will be lost between the script on the score and the script in the cast. As a result it may substitute a different script in its place, or just leave it blank. Director will try to resolve this problem (you'll notice that it asks you if it should adjust the scripts) but it rarely works if there have been non-trivial changes to the casts. In reality, it is usually easier to record what settings you had for the scripts originally and then to rebuild the application. If you have a very complex application that is not working in the new release, let us know and we can probably help you "port" it.

20. **When executing Lingo code, Director may not crash when encountering a bad line of code, it may simply exit the current function.** This can cause some confusing errors. You just wrote some new code and now the program runs, but with odd results. If you put in breakpoints you may realize that when it hits a line of code that has an error in it, instead of giving you a compile error early, or generating an error message at runtime, instead it simply breaks out of the current function and continues functioning. This means that you may not even be aware that whole sections of code are never executing.
21. **I want to do X with DART, but I don't see how.** Unfortunately, DART is supported by a small research grant, and has a small team of developers, so our documentation is pretty slim. Please avail yourself of all the resources we have created (the dart-users mailing list, the dart swiki, and soon the dart discussion groups). If it's possible, we will tell you how. If it's not, we'll tell you why.
22. **I want to use DART for a project, but I need it to do X.** The DART team is a research group, and is always open to collaboration. Our research interests are in AR/MR/ubicomp experience design and media theory, and are very broad. Joint projects that provide us funds to hire students or research scientists to add new features to DART are always of interest to us, especially if this happens in the context of an interesting research project. Furthermore, the IMTC group (one of the collaborators on DART) is a contract research organization; if you have something you want done, and have the money to pay for it, chances are we can do it! Just ask!

# APPENDIX F

# TEXT FILE FOR VIDEOTEXTURE TEMPLATE APP

This demo application shows the use of the VideoTexture.

Make sure that you have the cast files from DART/media/VideoContent

* What you have to do before starting the movie:

  - Nothing

* What you have to do while the movie runs:

  - If you have not already tried the Texture template app, run it first. All the other texture template apps are based off of this one.

  - Start the application

* What is supposed to happen:

  - This application demonstrates the video Texture component. In this example videotextures are placed on a HUDActor , and ObjectActor (the cube), and the ParticleActor. There is also a reflectionmap texture on the cube.

  - A video texture is seen on all three actors.

  - audio will also play synchronized with the video

  - if you are running Windows you can try out the fourth loop, which plays the video on a cube this time with the audio spatialized (currently spatial audio via OpenAL only works under Windows). The video uses a different cast this time (POVA_JRD_spatial) that contains a wav version of the audio track instead of to the quicktime version. The spatial audio will only work with mono wav files.

  - Move the camera around to see how the sound changes. There is a 3DAudioProperties script placed on the actor sprite. This is an optional script that lets you further tweak the 3D sound properties on an individual basis. Try changing settings in this script to hear how it affects the sound.

* Last time successfully played:

- April 7, 2006

- Maribeth @ cc.gatech.edu

# APPENDIX G

# TEXT FILE FOR WIZARDLINGO TEMPLATE APP

This example shows how wizard monitors Puppet side by using WizardLingo.

* What you need to do before starting the movie:

- Before running this application, two computers should have their own IP address.

- Run WizardLingo-Wizard.dir in the computer that will work as a wizard.

- Run WizardLingo-Puppet.dir in the computer that will work as a puppet.

- Currently, IP address of puppet is set as "199.77.129.125" in the wizard side.

If you want to use your own puppet's address, change the property of "Wizard of OZ" behavior in the wizard side.

* What you need to do after starting the movie:

- Move mouse pointer on the puppet screen.

* What is supposed to happen:

- Two texts having "WizardLingo" behavior will be updated by the mouse position of puppet side.

* Last time successfully tested:

- July 12, 2006

- Maribeth @cc.gatech.edu

# APPENDIX H

# STUDY PARTICIPANTS

Developer Groups, Descriptions, and Projects

| Participants | TECH | MANAGER | STUDENT | NOVICE | RENAISSANCE |
|---|---|---|---|---|---|
| Description | A computer scientist completing a PhD. in AR who extended DART | A computer professional, experienced with AR, who led the AR project for Duran Duran | A PhD student in Human Centered Computing who focused on AR research | A MS student with some software development experience who was inexperienced with AR | A Digital Media graduate student, with skills in 3D modeling and programming, who explores the medium of video games |
| Used DART? | Yes | No | No | Yes | Yes |
| Projects | A molecular modeling prototype and 3 experiment applications | AR system for a live performance. Collaborated with a team that included computer scientists, musicians, and crew | An AR installation set in a shipping container using handheld devices. Worked with a non-technologist collaborator. | An AR installation comprised of 4 AR puzzle kiosks and a center table that was a TUI. Collaborated with 3D artist. | An AR "toy" with a virtual dog and an AR art project where animated silhouettes in physical pictures frames would respond to each other. |

| Participants | MUSICIAN | PROFESSOR | ARTIST & RESEARCHER | DIRECTOR & DEVELOPER |
|---|---|---|---|---|
| Description | A researcher and artist who has built AR applications and AR authoring tools | A professor of digital media with extensive experience in the medium of AR | A professor and the technology manager of her lab focused on future cinema research and AR. | A theater director and the software developer that programmed in DART |
| Used DART? | Yes | No | Yes | Yes |
| Projects | Previously built music focused AR applications using the ARToolkit (and later TouchDesigner). He experimented with DART but did not use it for a full-fledged experience. | His students have built a variety of artistic AR experiences, often focused on historic tours | The research lab was originally built around DART. They used DART in multiple offerings of courses (used by ~45 students) and they built dozens of research projects/art installations using DART. | Designed, developed, and exhibited an AR version of a play |

# APPENDIX I

# INTERVIEWS CODING CHART

Occurrences of Guideline Related Comments in 11 Interviews

← more technical expertise — less technical expertise →

| Guideline | TECH | MANAGER | STUDENT | NOVICE | DEVELOPER | RENAISSANCE | RESEARCHER | MUSICIAN | PROFESSOR | ARTIST | DIRECTOR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| The need for rapid prototyping | x | x | x | x | x | x | x | x | | x | x |
| Early ideation artifacts into tech prototypes | | | x | | | | | | | | x |
| Support communication between stakeholders | | x | x | | x | | | x | | | x |
| The value of a layered dev environment | x | | x | x | x | | x | x | x | x | x |
| Mature content pipeline required | | x | | x | | x | x | x | | x | x |
| Access to diverse and arbitrary hardware | x | | | x | x | | x | | | x | x |
| Ability to swap between hardware technology | x | | | x | | | | | | | |
| Debugging support for entire system | | | x | x | x | | x | | | x | x |
| Supporting deployment (monitoring/debug) | x | x | | | x | | | | | | x |
| Formal evaluation support (logging/viz of data) | x | | | | | | | | | | |

Occurrences of DART Feedback in 8 Interviews

| DART Feedback | TECH | | | NOVICE | DEVELOPER | RENAISSANCE | RESEARCHER | MUSICIAN | | ARTIST | DIRECTOR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| General: Low barrier to entry | x | | | x | | | x | | | x | |
| General: Positive experience with DART | x | | | x | x | | x | | | x | x |
| Director Platform: had previous experience with Director (even if minimal) | | | | x | x | x | x | x | | x | |
| Director Platform: disliked use of Director and/or Lingo for DART | x | | | | | x | x | x | | x | |
| Director Platform: Liked use of timeline | x | | | x | x | | | | | | |
| Director Platform: Did not like use of timeline | | | | | | x | x | | | | |
| Director Platform: Need for code reuse | | | | | x | | x | | | | |
| Director Platform: Trouble loading 3D content | | | | x | | x | | | | | |
| Features: Used VideoActors | | | | x | x | | x | | | x | x |
| Features: Loading video tedious/frustrating | | | | | x | | x | | | x | x |
| Hardware: Liked hardware access | x | | | x | x | | x | | | x | x |
| Hardware: Used hardware swapping capabilities | x | | | x | | | | | | | |
| Debugging: Had problems identifying cause of errors | | | | x | x | | x | | | x | |
| Prototyping/Debug: Used DART for early prototyping | x | | | x | x | x | x | | | x | |
| Prototyping/Debug: Used Capture/Playback features | | | | x | | | | | | | |
| Prototyping/Debug: Used WoZ features | | | | x | | | | | | | |
| Prototyping/Debug: Didn't understand details or value of features | | | | | x | | x | | | x | x |
| Documentation: Wanted more complex example projects | | | | | | x | x | x | | x | x |
| Documentation: Wanted more detailed documentation | | | | | | x | x | x | | x | |
| Community: Got value from DART community | x | | | x | | x | x | | | x | x |

Occurrences of Authoring Needs in 11 Interviews

| Authoring Needs | TECH | MANAGER | STUDENT | NOVICE | DEVELOPER | RENAISSANCE | RESEARCHER | MUSICIAN | PROFESSOR | ARTIST | DIRECTOR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Current tools insufficient for their needs (no hardware access, complicated to use, constrained to application niche) | x | | | x | x | | x | | | x | x |
| Projects were/are affected by lack of understanding of affordances and constraints (by some members of team) | | x | x | x | x | x | x | | x | x | x |
| Ability to prototype without programming/scripting early on in the process | x | | x | | | | x | x | | x | x |
| Ability to transform low-tech artifacts into technology prototypes | | | x | | | | | | | | x |
| Prototyping activities focused on technology side | x | x | x | x | x | x | x | x | x | | |
| Prototyping activities focused on creative side | | | x | | | | x | | x | | x |
| Projects involved specialization of team members | | x | x | | x | | x | x | | x | x |
| Projects involved most members of team engaging with tech | x | | | x | | x | x | | x | x | |
| Felt exhausted, overwhelmed, burned-out after project(s) | | x | | x | | | | | | | x |
| Struggled with understanding all technology "beyond the software" that made up project | | | | x | x | | x | | x | x | x |

## REFERENCES


Abawi, D., Dorner, R., Haller, M., & Zauner, J. (2004, March 15). *Efficient Mixed Reality Application Development.* Paper presented at the 1st European Conference on Visual Media Production, London, England.

ARToolkit Mailing List Archive. (2000). Retrieved July 18, 2012, 2012, from http://www.hitl.washington.edu/artoolkit/mail-archive/search.php

Augmented Reality Software and Solutions by Total Immersion | Augmenting Your Reality. (2012). Retrieved July 3, 2012, 2012, from http://www.t-immersion.com/

Azuma, R. T. (1997). A Survey of Augmented Reality. *PRESENCE- Teleoperators and Virtual Environments, 6*(4), 355-385.

Barba, E., Rouse, R., Bolter, J. D., & MacIntyre, B. (2010). *Thinking Inside the Box: Meaning Making in a Handheld Augmented Reality Experience*. Paper presented at the The 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2010), Seoul, Korea.

Barker, S. (2006a). Retrieved from http://apollobeyond1.blogspot.com/

Barker, S. (2006b). Apollo Beyond Interactive Exhibit | Digital Frontiers Media Retrieved May 16, 2012, 2012, from http://digitalfrontiersmedia.com/apollo-beyond-interactive-exhibit

Barker, S., & Speckels, S. (2006, January 22, 2007). Apollo Beyond Retrieved May 16, 2012, 2012, from http://www.illustration4hire.com/apollobeyond/index.html

Bederson, B. B. (1995). *Audio Augmented Reality: a prototype automated tour guide*. Paper presented at the Conference Companion on Human Factors in Computer Systems, Denver, Colorado.

Bell, B., & Feiner, S. (2000, November 5-8). *Dynamic Space Management for User Interfaces.* Paper presented at the ACM Symposium on User Interface Software and Technology, San Diego, California.

Berry, R., Oikawa, M., Prasad, J., Unterberg, J., Liu, W., Cheok, A. D., & Kato, H. (2008). *Augmented Reality for Artists and Designers.* Paper presented at the ACM SIGGRAPH ASIA 2008: emerging technologies.

Billinghurst, M., Bowskill, J., Jessop, M., & Morphett, J. (1998, October 19-20). *A Wearable Spatial Conferencing Space.* Paper presented at the International Symposium on Wearable Computing, Pittsburgh, Pennsylvania.

Billinghurst, M., Weghorst, S., & Furness, T. A. I. (1998). Shared Space: An Augmented Reality Approach for Computer Supported Collaborative Work. *Virtual Reality, 3*, 25-36.

Cameron, C. (2012, May 24, 2010). Code-free Augmented Reality in Under 5 minutes [page linking to video of a developer building an AR application in Quartz Composer with no programming]. Retrieved from http://www.readwriteweb.com/archives/code-free_augmented_reality_in_under_5_minutes_video.php

Caudell, T. P., & Mizell, D. W. (1992, January). *Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes.* Paper presented at the Proceedings of Hawaii International Conference on System Sciences.

Chastine, J., Brooks, J. C., Zhu, Y., Owen, G. S., Harrison, R. W., & Weber, I. T. (2005, November 7-9). *AMMP-Vis: a collaborative virtual environment for molecular modeling.* Paper presented at the ACM Symposium on Virtual Reality Software and Technology, Monterey, California.

Chastine, J., & Zhu, Y. (2008). *The Cost of Supporting References in Collaborative Augmented Reality.* Paper presented at the Graphics Interface.

Chastine, J. W., Nagel, K., Zhu, Y., & Yearsovich, L. (2007). *Understanding the design space of referencing in collaborative augmented reality environments.* Paper presented at the Graphics Interface.

Cheok, A. D., Goh, K. H., Wei Lu, Farbiz, F., Fong, S. W., Teo, S. L., . . . Yang, X. (2004). Human Pacman: A Mobile, Wide-Area Entertainment System based on Physical, Social, and Ubiquitious Computing. *Personal and Ubiquitous Computing, 8*(2), 71-81.

Computers, A. (1992). *Inside MacIntosh: MacIntosh Toolbox Essentials* Addison-Wesley.

Conway, M., Audia, S., Burnette, T., Cosgrove, D., Christiansen, K., Deline, R., . . . Pausch, R. (2000). *Alice: Lessons Learned from Building a 3D System for Novices.* Paper presented at the CHI, The Hague, The Netherlands.

Detienne, F. (2001). *Software Design - Cognitive Aspects*. London.

Dey, A. K., Salber, D., & Abowd, G. D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human Computer Interaction*(2), 97-166.

Dow, S., Fortuna, J., Schwartz, D., Altringer, B., Schwartz, D. L., & Klemmer, S. R. (2011). *Prototyping Dynamics: Sharing Multiple Designs Improves Exploration, Group Rapport, and Results.* Paper presented at the Conference on Human Factors in Computing Systems (CHI '11), Vancouver, BC.

Dow, S., Lee, J., Oezbek, C., MacIntyre, B., Bolter, J. D., & Gandy, M. (2005a, Jun 15-17). *Exploring Spatial Narratives and Mixed Reality Experiences in Oakland Cemetery.* Paper presented at the ACM SIGCHI Conference on Advances in Computer Entertainment (ACE '05), Valencia, Spain.

Dow, S., Lee, J., Oezbek, C., MacIntyre, B., Bolter, J. D., & Gandy, M. (2005b). *Wizard of Oz interfaces for mixed reality applications.* Paper presented at the CHI '05 extended abstracts on Human factors in computing systems (CHI EA '05), Portland, Oregon.

Dow, S., MacIntyre, B., Gandy, M., & Bolter, J. D. (2004). *Prototyping Applications for the Physical World Using Integrated Capture/Playback Facilities.* Paper presented at the In Extended Abstracts of Conference on Ubiquitous Computing (UBICOMP04), Nottingham, U.K.

Dunser, A., Grasset, R., & Billinghurst, M. (2008). *A Survey of Evaluation Techniques Used in Augmented Reality Studies.* Paper presented at the International Conference on Computer Graphics and Interactive Techniques, Singapore.

echopai. (2007). AR--My Place Retrieved June 14, 2012, 2012, from http://www.youtube.com/watch?v=GyWveJ861zA

Edwards, K., Bellotti, V., Dey, A. K., & Newman, M. W. (2003). *Stuck in the Middle: The Challenges of User-centered Design and Evaluation for Infrastructure*. Paper presented at the the SIGCHI conference on Human factors in computing systems (CHI '03).

Feiner, S., MacIntyre, B., Haupt, M., & Solomon, E. (1993, November 3-5). *Windows on the World: 2D Windows for 3D Augmented Reality.* Paper presented at the ACM Symposium on User Interface Software and Technology, Atlanta, Georgia.

Feiner, S., MacIntyre, B., Hollerer, T., & Webster, T. (1997, October 13-14). *A Touring Machine: Prototyping 3D mobile Augmented Reality Systems for Exploring the Urban Environment.* Paper presented at the International Symposium on Wearable Computer, Cambridge, Massachussets.

Feiner, S., MacIntyre, B., & Seligmann, D. (1993). Knowledge-based Augmented Reality. *Communications of the ACM, 36*(7), 52-62.

FLARToolkit. (2011). Retrieved August 24, 2011, from http://www.libspark.org/wiki/saqoosha/FLARToolKit/en

Gandy, M., Catrambone, R., MacIntyre, B., Alvarez, C., Eiriksdottir, E., Hilimire, M., . . . McLaughlin, A. C. (2010, October 13-16). *Experiences with an AR Evaluation Test Bed: Presence, Performance, and Physiological Measurement.* Paper presented at the 2010 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Seoul, Korea.

Gandy, M., Jones, B., Robertson, S., O'Quinn, T., & Johnson, A. Y. (2009). *Rapidly Prototyping Marker Based Tangible User Interfaces.* Paper presented at the VMR '09 Proceedings of the 3rd International Conference on Virtual and Mixed Reality: Held as Part of HCI International 2009.

Gandy, M., MacIntyre, B., & Dow, S. (2004, Nov. 2-5). *Making Tracking Technology Accessible in a Rapid Prototyping Environment.* Paper presented at the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004), Arlington, Virginia.

Gandy, M., MacIntyre, B., Dow, S., & Bolter, J. D. (2006). Supporting Early Design Activities for AR Experiences. In M. Haller, M. Billinghurst & B. Thomas (Eds.), *Emerging Technologies of Augmented Reality: Interfaces and Design* (Vol. 1, pp. 160-180): IGI Global.

Gandy, M., MacIntyre, B., Presti, P., Dow, S., Bolter, J. D., Yarbrough, B., & O'Rear, N. (2005). *AR Karaoke: Acting in Your Favorite Scenes.* Paper presented at the International Symposium on Mixed and Augmented Reality, Vienna, Austria.

Greenberg, S., & Fitchett, C. (2001, November 11-14). *Phidgets: Easy Development of Physical Interfaces through Physical Widgets.* Paper presented at the 14th Annual ACM Symposium on User Interface Software and Technology, Orlando, Florida.

Grimm, P., Haller, M., Paelke, V., Reinhold, S., Christian Reimann, & Zauner, J. (2002, September 29). *AMIRE - Authoring Mixed Reality.* Paper presented at the The

First IEEE International Augmented Reality Toolkit Workshop, Darmstadt, Germany.

Guven, S., & Feiner, S. (2003, October 21-23). *Authoring 3D Hypermedia for Wearable Augmented and Virtual Reality.* Paper presented at the International Symposium on Wearable Computers, White Plains, New York.

Henderson, S., & Feiner, S. (2011, October). *Augmented Reality in the Psychomotor Phase of a Procedural Task.* Paper presented at the International Symposium on Mixed and Augmented Reality, Basel, Switzerland.

Hill, A., Barba, E., MacIntyre, B., Gandy, M., & Davidson, B. (2011, August 22-25). *Mirror Worlds: Experimenting with Heterogeneous AR.* Paper presented at the 2011 International Symposium on Ubiquitious Virtual Reality (ISUVR), Daejoen, South Korea.

Hohl, W. (2008). *Interactive Environments with Open-Source Software: 3D Walkthroughs and Augmented Reality for Architects with Blender 2.43, DART 3.0 and ARToolKit 2.72*: Springer Vienna Architecture.

Hollerer, T., Feiner, S., Terauchi, T., Rashid, G., & Hallaway, D. (1999). Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System. *Computers and Graphics, 23*(6), 779-785.

Kidd, C. D., Starner, T., Gandy, M., & Quay, A. M. (2000). The Beware Home: A Contextually Aware Haunted House *GVU Technical Report*. Atlanta: Georgia Institute of Technology.

Klemmer, S. R., Li, J., Lin, J., & Landay, J. (2004). *Papier-Mache: Toolkit Support for Tangible Input.* Paper presented at the ACM Conference on Human Factors in Computing Systems (CHI '04).

Klemmer, S. R., Newman, M. W., Farrell, R., Bilezikjian, M., & Landay, J. (2001). *The Designer's Outpost: A Tangible Interface for Collaborative Web Site Design.*

Paper presented at the UIST: ACM Symposium on User Interface Software and Technology.

Klemmer, S. R., Newman, M. W., Farrell, R., Meza, R., & Landay, J. (2000). A Tangible Evolution: System Architecture and Participatory Design Studies of the Designers' Outpost (CSD, Trans.). Berkely, CA: UC Berkeley.

Klemmer, S. R., Sinha, A. K., Chen, J., Landay, J., Aboobaker, N., & Wang, A. (2000). *Suede: A Wizard of Oz Prototyping Tool for Speech User Interfaces.* Paper presented at the UIST: ACM Symposium on User Interface Software and Technology.

Koc, A., & Cheng, L. (2011). The Coat Check/Le Vestiaire Emanethane (pp. An interactive augmented reality installation that focuses on the relationship between displacement, identity and technology). Toronto, ON.

Kooper, R., & MacIntyre, B. (2003). Browsing the Real-World Wide Web: Maintaining Awareness of Virtual Information in an AR Information Space. *International Journal of Human-Computer Interaction, 16*(3), 425-446.

Landay, J., & Myers, B. A. (2001). Sketching Interfaces: Toward a More Human Interface Design. *IEEE Computer, 34*(3), 56-64.

Lang, T. (2007). AR Second Life| AR Pit Retrieved November 8, 2011, from http://arsecondlife.gvu.gatech.edu/proj_pit.html

Lang, T., MacIntyre, B., & Zugaza, I. J. (2008). *Massively Multiplayer Online Worlds as a Platform for Augmented Reality Experiences.* Paper presented at the IEEE Virtual Reality Conference, Reno, Nevada.

Latulipe, C., Mann, S., Kaplan, C. S., & Clarke, C. L. A. (2006). *symSpline: Symmetric two-handed spline manipulation*. Paper presented at the Proceedings of the SIGCHI conference on Human Factors in computing systems, Montreal, Quebec, Canada.

Layar. 2011, from http://www.layar.com

Ledermann, F. (2011). APRIL - Augmented Presentation and Interaction Authoring Language Retrieved September 2, 2011, from http://studierstube.icg.tugraz.at/april/

lester. (2011, August 24, 2011). GE Augmented Reality Demo. Retrieved from http://www.augmentedplanet.com/2009/05/ge-augmented-reality-demo/

Li, Y., Hong, J. I., & Landay, J. (2007). Design Challenges and Principles for Wizard of Oz Testing of Location-Enhanced Applications. *IEEE Pervasive Computing, 6*(2), 70-75.

Li, Y., Hong, J. I., & Landay, J. A. (2004). *Topiary: a tool for prototyping location-enhanced applications*. Paper presented at the The 17th Annual ACM Symposium on User Interface Software and Technology (UIST '04).

Lima, A. J. R. d., Cunha, G. G., & Haguenauer, C. J. (2009). Descriptive Geometry Learning with the Aid of Augmented Reality. *Virtual Reality Journal, 2*(1).

Lin, J., Hong, J. I., & Landay, J. (2001). *DENIM: An Informal Tool for Early Stage Web Site Design*. Paper presented at the CHI '01 extended abstracts on Human factors in computing systems, Seattle, Washington.

Linden Lab: Makers of Second LIfe. (2011). Retrieved November 8, 2011, from lindenlab.com

Lyons, K., Gandy, M., & Starner, T. (2000, April 2000). *Guided by Voices: An Audio Augmented Reality System.* Paper presented at the International Conference on Auditory Display (ICAD) 2000, Atlanta, Georgia.

MacIntyre, B., Bolter, J. D., Vaughn, J., Hannigan, B., Gandy, M., Moreno, E., . . . Voida, S. (2003). *Three Angry Men: An Augmented-Reality Experiment in Point-of-View Drama.* Paper presented at the International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, Germany.

MacIntyre, B., Bolter, J. D., Vaughn, J., Hannigan, B., Moreno, E., Haas, M., & Gandy, M. (2002). *Three Angry Men: Dramatizing Point-of-View using Augmented Reality*. Paper presented at the ACM SIGGRAPH 2002 conference abstracts and applications (SIGGRAPH '02), San Antonio, Texas.

MacIntyre, B., & Feiner, S. (1996, November 6-8). *Language-level support for Exploratory Programming of Distributed Virtual Environments.* Paper presented at the ACM Symposium on User Interface Software and Technology, Seattle, Washington.

MacIntyre, B., & Feiner, S. (1998, July 19-24). *A Distributed 3D Graphics Library.* Paper presented at the SIGGRAPH, Orlando, Florida.

MacIntyre, B., Gandy, M., Dow, S., & Bolter, J. D. (2004, October 24-27). *DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences.* Paper presented at the The 17th annual ACM symposium on User interface software and technology (UIST '04), Santa Fe, New Mexico.

MacIntyre, B., Hill, A., Rouzati, H., Gandy, M., & Davidson, B. (2011, October 27-29). *Argon: An AR Web Browser Based on Open Standards.* Paper presented at the International Symposium on Mixed and Augmented Reality, Basel, Switzerland.

Mackay, W. E. (1998). *Video Prototyping: A technique for developing hypermedia systems*. Paper presented at the Conference Companion of ACM CHI '98 Human Factors in Computing Systems, Washington, D.C.

MacWilliams, A., Reicher, T., Klinker, G., & Bruegge, B. (2004). *Design Patterns for Augmented Reality Systems.* Paper presented at the International Workshop Exploring the Design and Engineering of Mixed Reality Systems, Funchal, Madeira.

MacWilliams, A., Sandor, C., Wagner, M., Bauer, M., Klinker, G., & Bruegge, B. (2003). *Herding Sheep: Live System Development for Distributed Augmented*

*Reality.* Paper presented at the International Symposium on Mixed and Augmented Reality.

Maes, P. (1995). Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *Communications of the ACM, 36*(7), 108-114.

McFerran, D. (2011). Hands On: Nintendo 3DS Augmented Reality Games. *Nintendo Life*. Retrieved from 3DS News @ Nintendo Life website: http://3ds.nintendolife.com/news/2011/03/hands_on_nintendo_3ds_augmented_reality_games

Merlin Mobility - Augmented Reality Instructions. (2011). Retrieved November 8, 2011, from http://www.merlinmobility.com

Metaio | Augmented Reality 3D. (2011). Retrieved November 8, 2011, from http://www.metaio.com/

Mobile Bristol | DShed. (2005). Retrieved June 18, 2012, 2012, from http://www.watershed.co.uk/dshed/mobile-bristol

Murphy, D. J., Kahari, M., & Ville-Veikko, M. (2010). *An Augmented Reality View on Mirror World Content with Image Space.* Paper presented at the IEEE Virtual Reality Conference.

Mynatt, E. D., Back, M., Want, R., & Frederick, R. (1997). *Audio Aura: Light Weight Audio Augmented Reality.* Paper presented at the ACM Symposium on User Interface Software and Technology.

Norton, M., & MacIntyre, B. (2005, October 5-8). *Butterfly Effect: An Augmented Reality Puzzle Game.* Paper presented at the International Symposium on Mixed and Augmented Reality, Vienna, Austria.

Oda, O., & Feiner, S. (2009, October). *Interference Avoidance in Multi-User Hand-Held Augmented Reality.* Paper presented at the International Symposium on Mixed and Augmented Reality, Orlando, Florida.
They did a research study on a new UI technique to help devices avoid each other. Written in Goblin XNA. The goals of Goblin are to facilitate 3D UI research in AR and VR.

Oda, O., Lister, L., & Feiner, S. (2008, January 8-10). *Developing an Augmented Reality Racing Game.* Paper presented at the International ICST Conference on Intelligent Technologies for Interactive Entertainment, Cancun, Mexico.

Okur, A., Ahmadi, S.-A., Bigdelou, A., Wendler, T., & Navab, N. (2011, October 26-29). *MR in OR: First Analysis of AR/VR Visualization in 100 intra-operative Freehand SPECT acquisitions.* Paper presented at the International Symposium on Mixed and Augmented Reality, Basel, Switzerland.

Olden, T., Bruin, T. d., & Kousemaker, D. (2004). Robotract Retrieved June 14, 2012, 2012, from http://blendid.nl/index.php?id=8

Oliver, J. (2008). Levelhead, November 8, 2011, from http://julianoliver.com/levelhead

Owen, C., Tang, A., & Xiao, F. (2003). *ImageTclAR: A Blended Script and Compile Code Development System for Augmented Reality*. Paper presented at the The International Workshop on Software Technology.

Pair, J., Chastine, J., & Gandy, M. (2002). *The Duran Duran Project: The Augmented Reality Toolkit in Live Performance.* Paper presented at the International Workshop on Augmented Reality Toolkit.

Piekarski, W. (2006). 3D Modelling with theTinmith Mobile Outdoor Augmented Reality System. *IEEE Computer Graphics and Applications, 26*(1), 14-17.

Piekarski, W., & Thomas, B. (2003, October 7-10). *An Object-Oriented Software Architecture for 3D Mixed Reality Applications.* Paper presented at the International Symposium on Mixed and Augmented Reality, Tokyo, Japan.

Presti, P., Gandy, M., MacIntyre, B., & Dow, S. (2005). *A Sketch Interface to Support Storyboarding of Augmented Reality Experiences.* Paper presented at the Conference on Computer Graphics and Interactive Techniques, Los Angeles, California.

punksmurfjie. (2006). Early AR Experiments, from http://www.youtube.com/watch?v=lls5fGJ-tOE

Qualcomm Augmented Reality SDK. (2011). 2011, from developer.qualcomm.com/dev/augmented-reality

Quinsland, K. (Writer). (2007). The Making of Woyzeck.

Read/Write World | A flexible fabric for exposing, connecting, and consuming geo-media and geo-data. (2011). Retrieved November 8, 2011, from http://readwriteworld.cloudapp.net/

Reitmayr, G., & Schmalstieg, D. (2001). *An open software architecture for virtual reality interaction*. Paper presented at the Proceedings of the ACM symposium on Virtual reality software and technology, Baniff, Alberta, Canada.

Rodriguez, R. (2005). Frank Miller's Sin City, DVD feature "The Robert Rodriguez 15 minute Flick School".

Rose, E., Breen, D., Ahlers, K. H., Crampton, C., Tuceryan, M., Whitaker, R. T., & Greer, D. S. (1995, June). *Annotating Real World Objects Using Augmented Reality.* Paper presented at the Computer Graphics International '95, Leed, United Kingdom.

Rosson, M. B., & Carroll, J. M. (1996). The Reuse of Uses in Smalltalk Programming. *ACM Transactions on Computer-Human Interaction, 3*(3), 219-253.

Roth, A. (2011). *The Arlab and Cave libraries: On authoring augmented reality and virtual reality experiences using a graphical programming language*. Paper presented at the 2011 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH).

Rothenstein, A. M., & Sizintsev, M. (2008). 52 Card Psycho: Implementation Details: York University.

Rouse, R. (2007). *Woyzek: Augmented Reality Performance Installation and Master's Project.* MA in Communication & Culture. Retrieved from http://www.rebeccarouse.com/uploads/5/7/8/1/5781195/woyzeckfinal_-_abcd.pdf

Rouse, R., Lee, M. M., Padgett, B., & Shepard, K. (2007). Woyzek: Augmented Reality Performance Installation (pp. An Augmented Reality installation presenting an adaptation of "Woyzek"). Toronto, ON.

Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavari, Z., Encarnacao, L. M., Gervautz, M., & Purgathofer, W. (2002). The Studierstube Augmented Reality Project. *PRESENCE- Teleoperators and Virtual Environments, 11*(1), 32-54
.

Schmalstieg, D., & Wagner, D. (2007, November 13-16). *Experiences with Handheld Augmented Reality.* Paper presented at the International Symposium on Mixed and Augmented Reality, Nara, Japan.

Seichter, H., Looser, J., & Billinghurst, M. (2008, 15-18 Sept. 2008). *ComposAR: An intuitive tool for authoring AR applications.* Paper presented at the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008. ISMAR 2008. .

Skolnik, M. R., & Roth, A. (2008). Multi-Sequential Poetry (pp. An augmented reality experience that makes a poem navigable and interactive). Toronto, ON.

SnapDragonAR from Future Stories. (2011). Retrieved May 16, 2012, 2012, from http://www.futurestories.ca/snapdragonar/

Sony EyePet. (2011). Retrieved November 8, 2011, from http://www.eyepet.com/

Spohrer, J. C. (1999). Information in Places. *Systems Journal, 38*(4), 602-628.

Stafford, A., & Piekarski, W. (2008). *User Evaluation of god-like interaction techniques.* Paper presented at the Australasian User Interface, Darlinghurst, Australia. talks about study done using TINMITH hardware (and software?)

State, A., Chen, D. T., Tector, C., Brandt, A., Chen, H., Ohbuchi, R., . . . Fuchs, H. (1994, October 17-21). *Case Study: Observing a Volume Rendered Fetus within a Pregnant Patient.* Paper presented at the IEEE Visualization '94, Washington, D.C.

State, A., Livingston, M. A., Hirota, G., Garrett, W. F., Whitton, M. C., Fuchs, H., & Pisano, E. D. (1996, August). *Techniques for Augmented-Reality Systems: Realizing Ultrasound-Guided Needle Biopsies.* Paper presented at the SIGGRAPH, New Orleans, Louisiana.

. String. (2012): String Labs Ltd. Retrieved from http://www.poweredbystring.com/

Su, L., & Feng, M. (2012). ARBox: Augmented Reality Experience Design. Atlanta. Retrieved from https://github.com/michaelfy/ARBox/wiki

Sutherland, I. (1965). *The Ultimate Display*. Paper presented at the IFIP Congress.

Tang, A., Owen, C., Biocca, F., & Mou, W. (2003). *Comparative Effectiveness of Augmented Reality in Object Assembly*. Paper presented at the ACM CHI '2003, Fort Lauderdale, Florida.

Thomas, B., Close, B., Donoghue, J., Squires, J., Bondi, P. D., Morris, M., & Piekarski, W. (2000). *ARQuake: An Outdoor/Indoor Augmented Reality First Person*

*Application*. Paper presented at the International Symposium on Wearable Computers, Atlanta, Georgia.

TOPPS 3D LIVE Online Augmented Reality Trading Card App. (2011). Retrieved November 8, 2011, from http://www.topps.com/discover/games-and-apps/topps-3d-live

Unity AR Toolkit (UART). (2011). Retrieved August 24, 2011, from https://research.cc.gatech.edu/uart/

UNITY: Game Development Tool. (2011). Retrieved November 8, 2011, from unity3d.com

Voices of Oakland. (2011). Retrieved November 8, 2011, from http://argon.cc.gatech.edu/oakland_project.html

VRML Virtual Reality Modeling Language. (2011). Retrieved November 8, 2011, 2011, from http://www.w3.org/MarkUp/VRML/

Vuzix - View the Future Today. (2011). Retrieved November 8, 2011, from http://www.vuzix.com/home/

Wagner, I., Broll, W., Jacucci, G., Kuutii, K., McCall, R., Morrison, A., . . . Terrin, J. J. (2009). On the Role of Presence in Mixed Reality. *PRESENCE- Teleoperators and Virtual Environments, 18*(4), 249-276.

Walker, D. (2009). The Augmented Cambodian (pp. An augmented reality project that tells the visually compelling story of, Salao Mao, a Cambodian artist).

Warne, P., & Wozniewski, M. (2011). Farrago: AR Funkit for iPhone and iPod Touch: Hololabs Studio. Retrieved from http://farragoapp.com/

Wikipedia, T. F. E. (2011). Eye of Judgement. Retrieved from http://en.wikipedia.org/wiki/The_Eye_of_Judgment

Wikitude. 2011, from http://www.wikitude.org

Wolff, E. (2004, November 1, 2004). Tool Time at Pixar. *Millimeter, The Professional Resource for Production and Post*.

Yan, X., Barba, E., Radu, I., Gandy, M., Shemaka, R., Schrank, B., . . . Tseng, T. (2011, 26-29 Oct. 2011). *Pre-patterns for designing embodied interactions in handheld augmented reality games.* Paper presented at the Mixed and Augmented Reality - Arts, Media, and Humanities (ISMAR-AMH), 2011 IEEE International Symposium On.

Zauner, J., & Haller, M. (2004, June). *Authoring of Mixed Reality Applications Including Multi-Marker Calibration for Mobile Devices.* Paper presented at the 1oth Eurographics Symposium on Virtual Environments.

# VITA

## Maribeth Gandy Coleman

Maribeth was born in Mobile, Alabama. She received a B.Cmpe in Computer Engineering from Georgia Tech in 1998 and an M.S. in Computer Science in 2000. In 2000 she accepted a research scientist position at Georgia Tech and then resumed her graduate studies as a part-time Ph.D. student in Computer Science in 2003. When she is not working on her research, Maribeth enjoys running, cooking for friends, playing RockBand, and caring for her numerous pets.